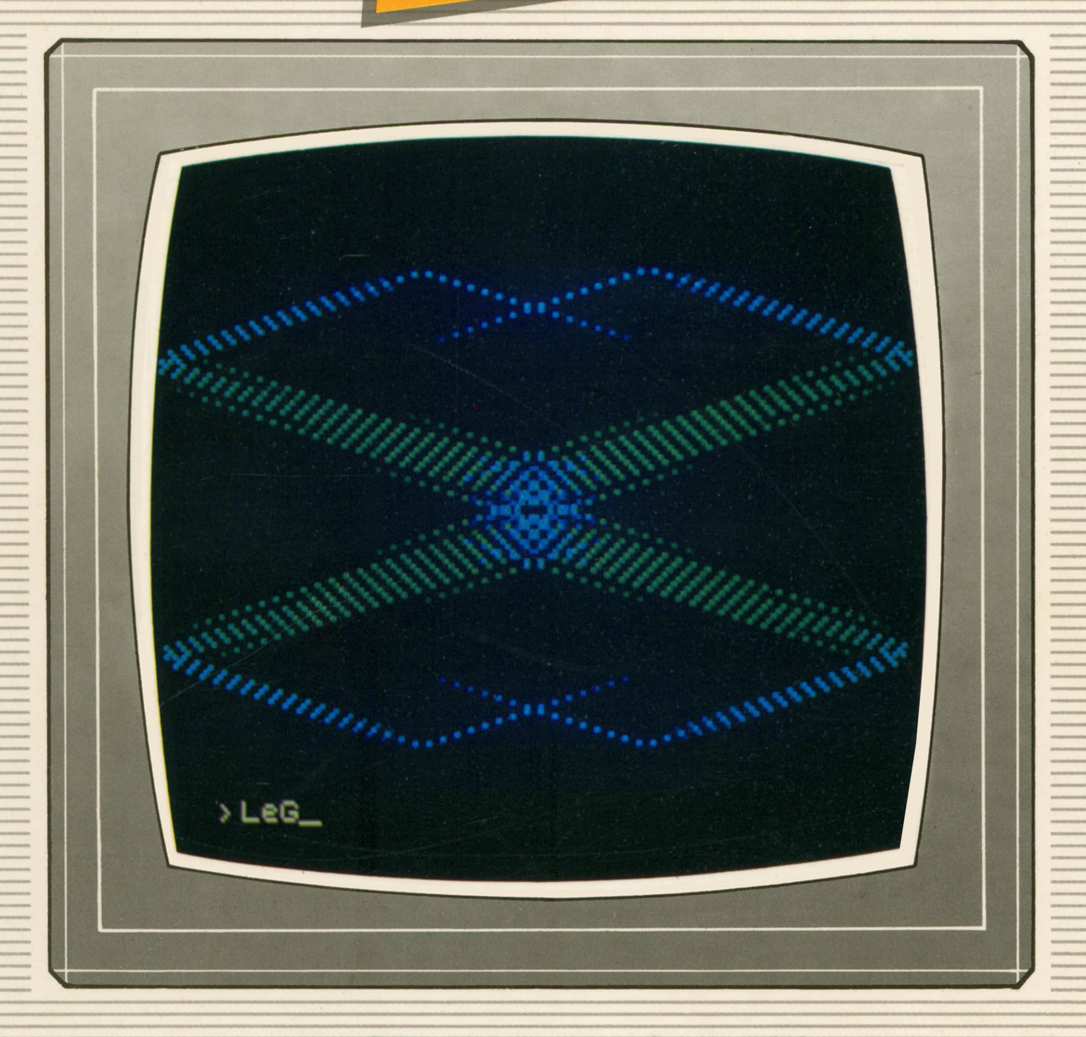
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 40

PASO A PASO



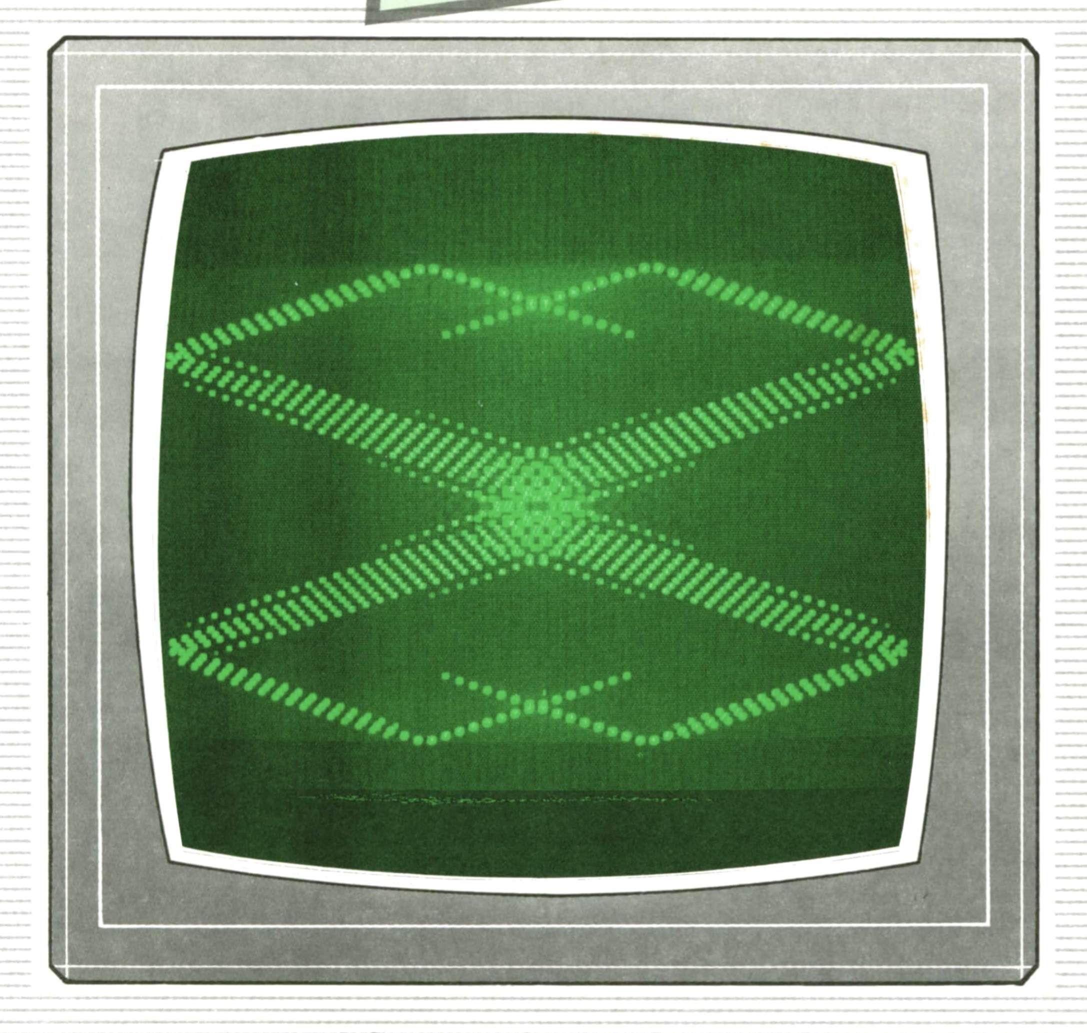
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 40

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Jorge Thomas, Técnico en Informática. OTROS LENGUAJES (ADA): Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 455 09 99. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-191-6
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.
Agosto, 1988.
P.V.P. Canarias: 335,–.

INDICE

4 INFORMATICA BASICA

8 MAQUINA 8088

14 PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

29 TECNICAS DE ANALISIS

31 TECNICAS DE PROGRAMACION

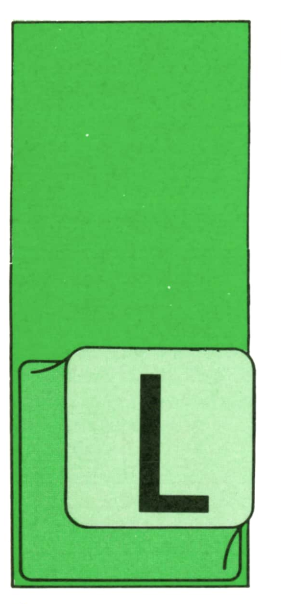
34 APLICACIONES

38 OTROS LENGUAJES

LA INFORMATICA BASICA

LAS APLICACIONES DE LA TELEINFORMATICA

Las redes telemáticas



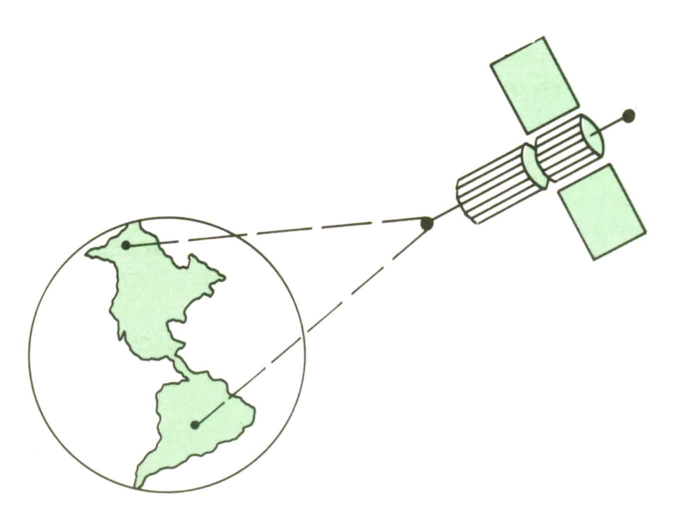
LAS redes telemáticas son el producto de unir los servicios informáticos con los medios de comunicación. Tanto las diferentes formas de conexión de los equipos con las líneas como las dimensiones de dichas redes pueden ser de diferentes tipos. En principio podemos afirmar que estas redes no tienen por qué tener dimensiones enormes; se pueden limitar a zonas locales o áreas concretas.

En Estados Unidos se estableció una red de más de un centenar de ordenadores de distinta capacidad para enlazar Universidades y centros de investigación que precisaban la misma información.

En principio, la red estaba compuesta por la conexión de un grupo de ordenadores encargados de enlazar y comunicar una serie de centros con un segundo bloque encargado de proporcionar la información. Hoy la actividad ha rebasado el ámbito de los Estados Unidos y es una gran red intercontinental.

Además de este tipo de redes existen **redes unilaterales** de distribución de televisión a través de microondas y, como sistema cableado, el teléfono. **Redes interactivas** son aquellas en las que la comunicación ya no es unilateral, como ejemplo está la distribución de las emisiones de televisión mediante cable coaxial, que da la posibilidad de «intervenir» al espectador en los «programas».

Con la utilización de la fibra óptica para la distribución de comunicaciones, el número de canales se hace enorme, por lo que se decide su utilización en la compartición, ya no de programas televisivos, sino de informaciones almacenadas en ordenadores situados en locales públicos de gestión, Universidades, bancos, etc. Con esta aplicación el abonado de este servicio podrá desde su domicilio realizar gestiones, solicitar información, reservar plazas en hoteles, ser chequeado por un determinado centro médico, etc. En Francia se multiplican ya los sistemas de redes locales para el diagnóstico a distancia, especialmente útil para personas de tercera edad. Otras redes más complejas incorporan monitores de vídeo para la observación del enfermo.

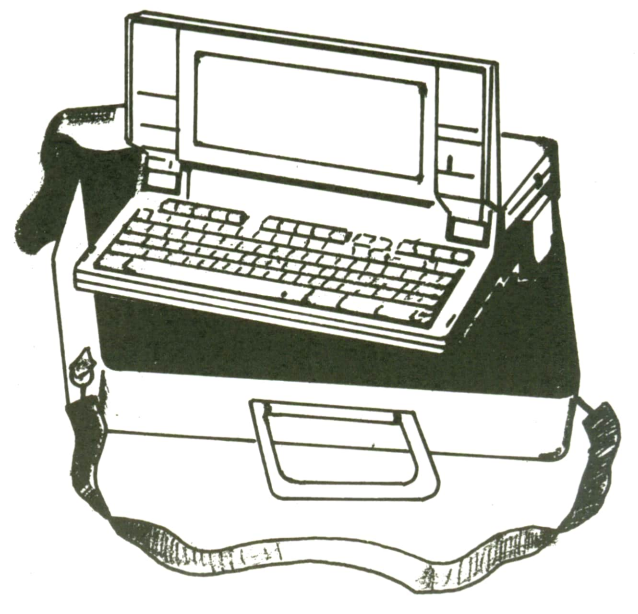




Las reales telemáticas mundiales garantizan la comunicación instantánea entre ordenadores distantes.



Existen terminales transportables, también aplicables a la ofimática.



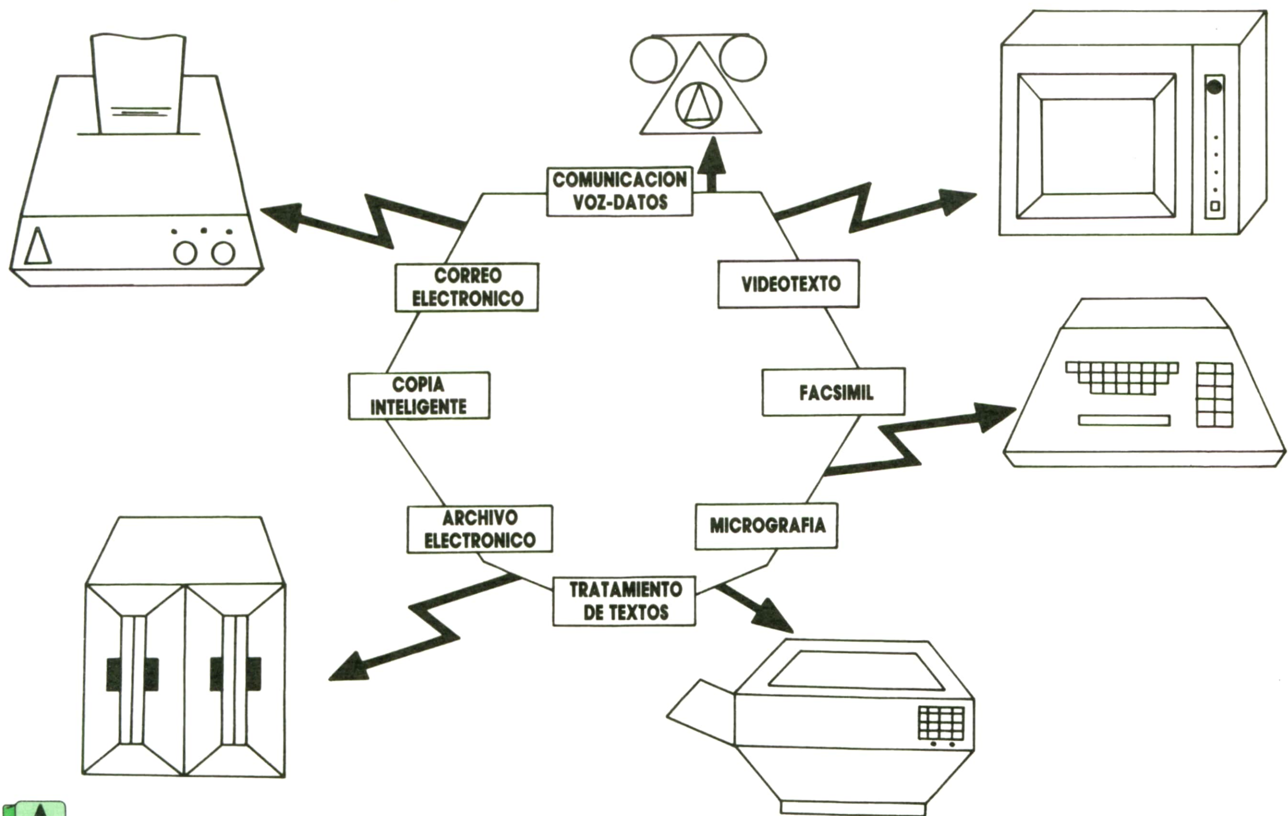


El teletrabajo

Consiste en poder realizar el trabajo diario de oficina en otro lugar; para ello se necesita disponer de medios de comunicación. El teletrabajo es posible, principalmente, en el sector terciario, donde se unen los medios informáticos y telemáticos.

Existen ya muchas experiencias en este campo y cada día se desarrolla más.

Los **medios** que necesita el teletrabajo son:



Integrando todas las tareas específicas de las oficinas se pueden automatizar todos los trabajos.

— Redes de transmisión de datos.
— Teléfonos y sus accesorios (contestador automático, registrador, etc.).
— Télex.
— Equipos de tratamiento de textos.
— Videotext y teletext.
— Terminales de pantalla y otros equipos periféricos.
— Telecopiadoras.

Los locales en los que deben ubicarse estos medios son:

El domicilio del empleado. El equipo puede pertenecer al empleado, con lo cual tiene la posibilidad de poder conectarse con las empresas en las que es contratado. En otros casos es la empresa la que proporciona el sistema.

El telelocal. Locales de una o varias empresas equiparados con materiales convenientes para acercar sus servicios a los clientes.

Actualmente las tareas que es posible realizar son:

— Tareas de tratamiento de textos: composición, corrección, difusión...
— Tareas de gestión administrativa. Consulta y mantenimiento de archivos y otras tareas que actualmente se realizan a través de terminales.
— Tareas que implican relación con el público. Consultas a bases de datos, reservas, pedidos, inscripciones, telepagos...
— Tareas informáticas. Especialmente programación.



El sistema teletext

Los inicios de este sistema se fechan en la década de los sesenta, en Gran Bretaña, donde se investigaba en cómo incluir subtítulos en las imágenes de televisión.

A principios de los setenta, en Francia se estudiaba la viabilidad del teletext. En 1980, en el Reino Unido, había 400.000 receptores de teletext en la red nacional de televisión.

Característica	Teletexto	Videotexto
Interacción usuario-ordenador del sistema	Unidireccional (emisión)	Bidireccional
Medio de transmisión	Señal de TV	Líneas telefónicas
Capacidad	Varios cientos de páginas en canal compartido. Aproximadámente 40 000 si se utiliza todo el canal.	Teóricamente sin límite, sólo depende de la capacidad de almacenamiento del medio
Tiempo de espera del usuario	Un promedio de 13 segundos por revista de 100 páginas; el tiempo de acceso es independiente del número de usuarios	Un ordenador como el DEC11/70 da servicios a unos 200 usuarios con tiempo de espera de varios segundos. Para mayor número de usuarios hay que aumentar la potencia del ordenador si se quiere conservar el mismo tiempo de espera.
Contenido de la información	Noticias, parte meteorológico, de interés general	Enciclopédica
Coste	Sin cargo para el usuario; el servicio lo pagan los anunciantes	Cargo en función de los servicios utilizados
Condiciones de acceso	Con decodificador especial; se puede utilizar en televisión de circuito cerrado y en televisión por cable	Se puede utilizar palabra clave de acceso para limitar la utilización de algunas bases de datos, por ejemplo, los informes confidenciales de una compañía.
Correo electrónico	No es posible	Posible: usuario a usuario o usuario a fuente de información. Se puede efectuar comunicación interredes con protocolos compatibles como el X.25.
Ampliación del sistema	Sólo se necesita un ordenador por sistema de teletexto; número ilimitado de usuarios	El número de ordenadores del sistema depende del número de usuario.


Cuadro comparativo entre las características del teletexto y el videotexto.

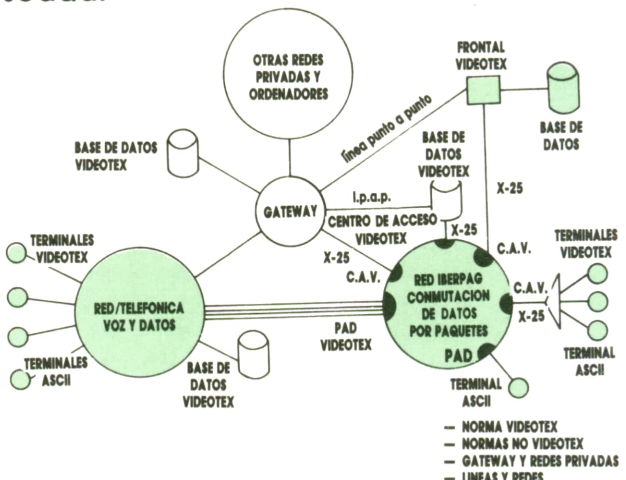
En la figura podemos apreciar los distintos elementos que constituyen el sistema teletext. La información que se distribuye a través del sistema de teletexto se compone de un terminal en edición. La información en el ordenador se transmite mediante alguna de las líneas no utilizadas de la imagen del televisor. La información de teletext aparece en la pantalla como texto gráfico, o una combinación de ambos.

El usuario utiliza el teclado numérico del sistema teletext para seleccionar entre varios cientos de páginas la que desea visualizar. Las páginas se emiten de modo continuo y secuencialmente, es decir, una vez terminado el texto se vuelve a empezar desde la primera página.



El sistema videotext

En la década de los setenta se desarrolló una tecnología complementaria al teletext llamada videotext (viewdata). La principal diferencia entre el teletext y el videotext es el modo en que se accede a la información. Ambos usuarios disponen de un teclado numérico por el que seleccionan las páginas deseadas. Cuando el usuario de teletext pulsa los botones para seleccionar una página determinada, programa el terminal para que espere la emisión de la página deseada.

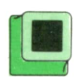



Configuración general de un Sistema Videotex.

Sin embargo, cuando el usuario del videotext pulsa los botones del teclado, sus instrucciones pasan al ordenador del sistema; el ordenador encuentra la página seleccionada y envía la información al usuario. Por tanto, en el sistema teletext la comunicación es unidireccional, ya que el usuario ha de esperar su turno de emisión; en contraposición, el videotext es bidireccional, ya que la fuente responde directamente a las órdenes del usuario.

El sistema videotext utiliza como terminales los aparatos de televisión de uso doméstico, y accede, a través de las líneas telefónicas, a base de datos. El sistema se realiza de forma que unos abonados determinados, llamados suminis-

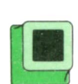
tradores, ponen sus informaciones a disposición de las personas que tengan teléfono. El suministrador fija el contenido, el coste y pone a ciertos usuarios restricciones en el acceso a la información.



Ventajas e inconvenientes del videotext

Las ventajas del videotext frente a la informática tradicional son claras: interconexión universal entre terminales, acceso también universal a los centros de servicio conectados al sistema, interfaz de fácil uso, procedimientos de comunicación normalizados, costes menores de terminal y de comunicaciones.

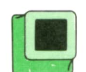
En cuanto a los inconvenientes, aún hoy siguen siendo bastante serios: tiempos de respuesta lentos, incapacidad de soportar aplicaciones en tiempo real y de gran volumen de información, mayor coste del software de estos sistemas frente a los tradicionales.



Sistemas de comunicación de oficinas

Mediante los sistemas de comunicación se ofrece una amplia gama de servicios de voz, datos, de alta y baja velocidad, redes de datos públicas y privadas. Al igual que se pueden compartir otro tipo de recursos como: impresoras, sistemas de archivo, procesadores de aplicaciones, bases de datos y memorias de masa.

En cuanto a servicios externos que se pueden proporcionar mediante conexiones apropiadas, están el correo electrónico, proceso de textos, el videotext y teletext.



Aplicaciones domésticas

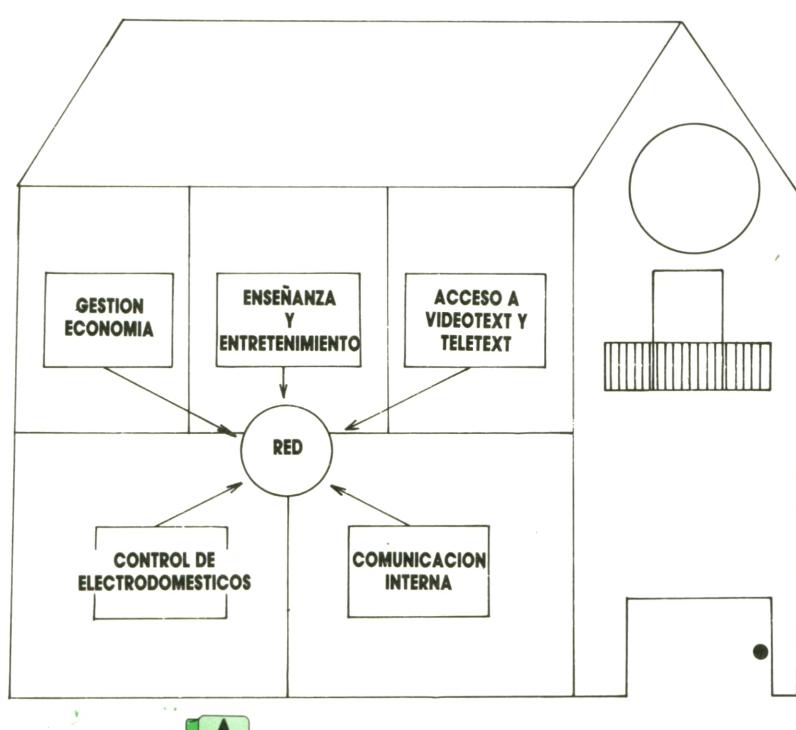
La forma de realizar un sistema en el hogar dependerá en gran medida de los terminales de los que se disponga y los medios de transmisión, tanto a nivel interno como externo.

La aplicación de la información a nivel doméstico es un tema muy atractivo y se prevé un fuerte desarrollo en poco tiempo. Actualmente en países como Estados Unidos, las aplicaciones son cada vez más numerosas; entre ellas se conocen:

— La transmisión interior de señales de vídeo, sonido, voz y datos.

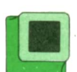
— Conexión con diversos equipos de electrodomésticos, desde dentro y fuera de la casa.

— Sistemas de entretenimiento y educación, etc.

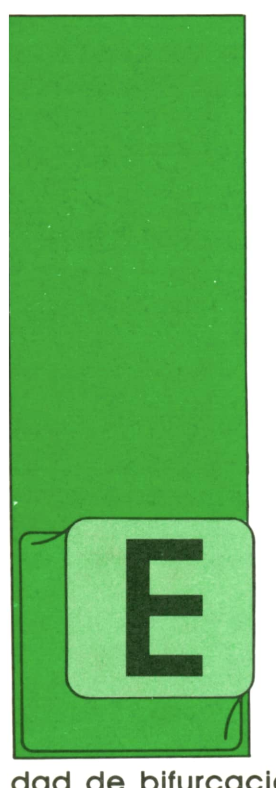


Informática en el hogar.

MAQUINA 8088



Instrucciones de iteración



EXISTEN tres instrucciones que facilitan la programación de bucles basándose en la utilización como contador de un registro específico, el registro CX. Se trata en realidad de bifurcaciones condicionadas al valor del registro CX. Cada vez que se ejecuta una de estas instrucciones se disminuye en una unidad el valor del contador contenido en CX y se bifurca (para continuar el bucle) si se cumple una condición específica de cada instrucción. En caso contrario, la ejecución continúa con la instrucción siguiente, lo que implica normalmente la terminación del bucle.

```
[etiqueta:]   LOOPXX   objetivo
```

Etiqueta. Es un nombre opcional.

LOOPXX: Representa a cinco nemotécnicos de las tres instrucciones citadas: LOOP, LOOPE, LOOPZ, LOOPNE y LOOPNZ, cuyas condiciones de bifurcación son las siguientes:

— LOOP bifurca al objetivo si CX es diferente de cero.

— LOOPE y su nemotécnico equivalente LOOPZ bifurcan al objetivo si CX es diferente de cero y el «flag» ZF está en «on» (valor 1).

— LOOPNE y su nemotécnico equivalente LOOPNZ bifurcan al objetivo si CX es diferente de cero y el «flag» ZF está en «off» (valor 0).

Objetivo. Es el operando mediante el cual se especifica la dirección a la que debe transferirse la ejecución. El objetivo de estas instrucciones tiene que ser SHORT, es decir, debe corresponder a una posición de memoria que no esté situada a una distancia superior a 127 bytes de la instrucción LOOPXX.

Pertenece también a este grupo la instrucción JCXZ, cuyo nemotécnico es una abreviatura de «jump if CX is zero», que significa «salto si CX es cero». Esta instrucción comparte la característica principal del grupo, es decir, efectúa una bifurcación a un objeto tipo SHORT, que está condicionada al valor de CX. Se diferencia, sin embargo, en que no decrementa en una unidad el contenido de CX cada vez que se ejecuta.



Números enteros

El 8088 puede operar con números enteros codificados en decimal y con números enteros codificados en binario. Por razones de extensión y utilidad vamos a limitarnos a estos últimos.

Los números binarios pueden tener una longitud de 1 ó 2 bytes (8 ó 16 bits) que permiten diferenciar 256 ó 65536 números, respectivamente. Dichos números se pueden representar de dos formas distintas, según se vayan a utilizar o no números negativos.

Cuando se desean manejar números que pueden ser positivos y negativos, se utiliza una representación en la que un bit se dedica al signo. Cuando se van a manejar números que no van a ser nunca negativos, todos los bits se emplean para representar la magnitud del número. Los primeros se denominan «binarios con signo» y los segundos «binarios sin signo». Resultan, por tanto, cuatro clases de números enteros binarios. En la siguiente tabla se expresan los números que se abarcan en cada clase:

	binarios sin signo	binarios con signo
Longitud 1 byte	del 0 al 255	del -128 al +127
Longitud 2 bytes	del 0 al 65535	del -32768 al +32767



Flags aritméticos

En las operaciones aritméticas de números enteros puede darse el caso de que el resultado de la operación se salga de los límites de la clase a la que pertenecen los operandos. Para detectar estas situaciones excepcionales, el 8088 dispone de un conjunto de «flags» o indicadores, donde cada vez que se efectúa una operación aritmética, se refleja información acerca de cómo se ha desarrollado la operación.

Cada flag, que se guarda en un bit, puede estar en dos estados: activado (valor 1) y desactivado (valor 0).

Como se vio en el tomo 37, las instrucciones de transferencia condicional permiten bifurcar a otras partes de un programa, dependiendo del estado de dichos «flags».

Los «flags» relacionados con las operaciones aritméticas son los siguientes:

— Flag de acarreo CF («Carry Flag»). Este flag se activa después de instrucciones de suma o resta en la que el resultado ha perdido el bit más significativo por haber excedido la capacidad del registro que lo contiene (incluyendo el bit de signo). En todos los demás casos este flag queda desactivado (valor 0). Por tanto, este flag sirve para detectar resultados anómalos en operaciones de binarios sin signo y puede ser ignorado cuando se opere con binarios con signo.

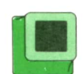
— Flag de «overflow» OF («Overflow Flag»). Este flag se activa después de instrucciones de suma o de resta en la que el resultado ha perdido el bit más significativo por haber excedido la capacidad del registro que lo contiene (excluyendo el bit de signo). En todos los demás casos este flag queda desactivado. Por tanto, este flag sirve para detectar resultados anómalos en operaciones de binarios con signo y puede ser ignorado cuando se opere con binarios sin signo.

— Flag de signo SF («Sign Flag»). Después de cada operación aritmética o lógica, este flag adquiere siempre el valor del bit de signo del resultado. Puede ignorarse en las operaciones de binarios sin signo.

— Flag de resultado igual a cero ZF («Zero Flag»). Este flag se activa siempre que el resultado de una operación aritmética o lógica sea 0 y se desactiva en los demás casos.

— Flag auxiliar de acarreo AF («Auxiliary carry Flag»). Se activa siempre que, en una suma, el byte menos significativo del resultado se haga mayor de FF o, en una resta, se haga menor de 00. Tiene interés sólo en las operaciones con números codificados en decimal.

— Flag de paridad PF («Parity Flag»). Se activa cuando en el byte menos significativo del resultado hay un número par de bits con valor 1. Tiene interés sólo para comprobar la corrección de códigos ASCII.



Instrucciones aritméticas

Vamos a tratar a continuación de un conjunto de instrucciones que permiten efectuar operaciones aritméticas con números enteros binarios.

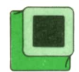
Es importante resaltar que en las operaciones de multiplicación y división hay que distinguir cuándo se están empleando números binarios «con signo» y «sin signo», ya que la configuración de bits del resultado depende de dicha circunstancia. Por esta razón, las instrucciones para multiplicar y dividir binarios sin signo (MUL y DIV) son diferentes de las que se emplean con los binarios con signo (IMUL e IDIV).

Sin embargo, esta cuestión no afecta a las deas operaciones aritméticas en las que el resultado es el mismo, independientemente de que los operandos se interpreten como binarios «con signo» o «sin signo».

La única distinción debe hacerse en las situaciones excepcionales en las que el resultado se salga fuera de los límites de cada clase. Como se ha explicado anteriormente, los flags CF y OF permiten detectar estas situaciones y las instrucciones JC, JO e INTO permiten bifurcar a la parte de programa donde se realizan las acciones correctoras necesarias.

Antes de pasar a explicar las distintas instrucciones conviene recordar que las

referencias a memoria admiten todas las variantes que ya se explicaron en el apartado «Referencias explícitas a la memoria» del tomo 13.



Sumas, restas y comparaciones

Las tres instrucciones que se estudian a continuación actualizan el mismo conjunto de flags (CF, OF, SF, ZF, AF y PF) y tienen la misma sintaxis:

```
[etiqueta1:]    ADD    destino,origen
[etiqueta2:]    SUB    destino,origen
[etiqueta3:]    CMP    destino,origen
```

— «destino» puede representar un registro (que no sea de segmento) o una posición de memoria.

— «origen» puede representar: un operando inmediato, un registro (que no sea un segmento) o una posición de memoria si el destino es un registro.

— ADD es el nemotécnico de la instrucción de suma, la cual hace que el operando «origen» se sume con el operando «destino» y que el resultado se guarde en el propio «destino».

— SUB es el nemotécnico de la instrucción de sustracción, la cual hace que el operando «origen» se reste del operando «destino» y que el resultado se guarde en el propio «destino».

— CMP es el nemotécnico de la instrucción de comparación, la cual hace que el operando «origen» se reste del operando «destino» pero sin que el resultado se guarde en ningún sitio, ya que el objetivo de esta instrucción es la actualización de los flags, de forma que reflejen la comparación de los operandos, para a continuación efectuar alguna transferencia condicional con alguna instrucción del tipo JXXX (ver tomo 37).



Incrementos, decrementos y cambios de signo

Hay tres instrucciones que permiten modificar el valor binario de un registro o posición de memoria, sumándole o restándole una unidad o cambiándole de signo. Las dos primeras instrucciones actualizan los flags OF, SF, ZF, PF y AF y la tercera actualiza además el flag CF. Se escriben de la siguiente forma:

```
[etiqueta1:]    INC    destino
[etiqueta2:]    DEC    destino
[etiqueta3:]    NEG    destino
```

— «destino» representa al registro o la posición de memoria que va a ser modificada.

— INC es el nemotécnico (abreviatura de «Increment») que provoca la adición de una unidad al destino.

— DEC es el nemotécnico (abreviatura de «Decrement») que provoca la sustracción de una unidad al destino.

— NEG es el nemotécnico (abreviatura de «Negate») que provoca el cambio de signo del destino. Esta instrucción presupone que el número contenido en el destino es un binario con signo y lo sustituye por el resultado de restar dicho número de cero.



Multiplicaciones

En la operación de multiplicación hay que distinguir si los operandos son binarios «con signo» o «sin signo», por esta razón, existen dos instrucciones diferentes MUL e IMUL.

Estas instrucciones se caracterizan por tener un solo operando que actúa como uno de los factores en la multiplicación; el otro factor debe estar previamente cargado en el acumulador. El resultado se obtiene con doble precisión (doble número de bits).

Cuando los factores son bytes, el resultado se devuelve en dos bytes, el byte menos significativo en AL (acumulador) y el más significativo en AH (extensión del acumulador). Cuando los factores son palabras, el resultado se devuelve en dos palabras, la palabra menos significativa en AX (acumulador) y la más significativa en DX (extensión del acumulador).

A su terminación, ambas instrucciones dejan los flags: AF, PF, SF y ZF en un estado indefinido.

La sintaxis es la siguiente:

```
[etiqueta1:]    MUL    origen
[etiqueta2:]    IMUL   origen
```

— «origen» representa el registro o la posición de memoria que contiene uno de los factores de la multiplicación.

— MUL es el nemotécnico (abreviatura de «Multiply») de la instrucción de multiplicación de binarios sin signo.

— IMUL es el nemotécnico (abreviatura de «Integer Multiply») de la instrucción de multiplicación de binarios con signo.

Divisiones

En las divisiones también hay que distinguir si los operandos son binarios «con signo» o «sin signo», y, por tanto, existen dos instrucciones que diferencian ambos casos; DIV e IDIV.

Las dos instrucciones se caracterizan por tener un solo operando que actúa como el divisor en la división. El dividendo debe definirse con doble precisión que el divisor y debe estar previamente cargado en el acumulador (la mitad menos significativa) y en la extensión del acumulador (la mitad más significativa».

En las divisiones de bytes el acumulador es el resgistro AL y su extensión es el AH. En las divisiones de palabras el acumulador es el registro AX y su extensión es el DX.

Al terminar la división el cociente queda en el acumulador y el resto en su extensión, y los flags CF, OF, AF, PF, SF y ZF quedan indefinidos.

Estas instrucciones se escriben de la siguiente forma:

```
[etiqueta1:]   DIV    origen
[etiqueta2:]   IDIV   origen
```

— «origen» representa el registro o la posición de memoria que contiene el divisor de la división.

— DIV es el nemotécnico (abreviatura de «Divide») de la instrucción de división de binarios sin signo.

— IDIV es el nemotécnico (abreviatura de «Integer Divide») de la instrucción de división de binarios con signo.

CBW y CWD son dos instrucciones auxiliares, que se pueden emplear antes de ejecutar una IDIV, para duplicar la precisión del número contenido en el acumulador, extendiendo su signo a la extensión del acumulador.

CBW (iniciales de «Convert Byte to Word») convierte el número contenido en AL en un número contenido en AX.

CWD (iniciales de «Convert Word to Doubleword») convierte el número contenido en AX en un número contenido en la pareja de registros AX, DX.



Instrucciones lógicas

Pertenecen a este grupo cuatro instrucciones que efectúan las operaciones booleanas: «no», «y», «o» y «o exclusivo» y una quinta instrucción (TEST) que sirve para comparar y que (igual que CMP) no modifica los operandos.

La negación lógica se ejecuta con la instrucción NOT, que no altera el estado de los flags y cuyo formato es:

```
[etiqueta:]  NOT  destino
```

— «destino» puede ser registro o una posición de memoria.

— NOT es un nemotécnico de la instrucción que efectúa la negación lógica del contenido del «destino», es decir, cambia ceros por unos y unos por ceros.

Las demás instrucciones lógicas AND, OR, XOR y TEST funcionan con dos operandos, el primero denominado «destino» y el segundo «origen». Aplican reglas específicas a las parejas de bits extraídas de ambos operandos y (exceptuando la TEST) guardan el resultado en el destino.

Los flags resultan afectados de la forma siguiente: CF y OF quedan siempre desactivados, AF queda indefinido y SF y ZF quedan actualizados de acuerdo con su propio significado (según se ha explicado anteriormente).

Los formatos de estas instrucciones son los siguientes:

```
[etiqueta1:]  AND   destino,origen
[etiqueta2:]  OR    destino,origen
[etiqueta3:]  XOR   destino,origen
[etiqueta4:]  TEST  destino,origen
```

— «destino» puede representar un registro (que no sea de segmento) o una posición de memoria.

— «origen» puede representar: un operando inmediato, un registro (que no sea de segmento) o una posición de memoria si el destino es un registro.

— AND efectúa la operación lógica «y». Un bit del resultado toma el valor 1 sólo si los dos bits de los operandos son 1.

— OR efectúa la operación lógica «o». Un bit del resultado toma el valor 1 sólo si alguno de los bits de los operandos son 1.

— XOR efectúa la operación lógica «o exclusiva». Un bit del resultado toma el valor 1 si los bits de los operandos son diferentes.

— TEST funciona igual que la instrucción AND, pero sin guardar el resultado en el destino. Se emplea para comparar datos y realizar transferencias condicionales de forma análoga a la instrucción aritmética CMP.



Desplazamientos

Hay cuatro instrucciones que permiten desplazar hacia la izquierda o hacia la derecha los bits contenidos en un registro o una posición de memoria.

Estas instrucciones admiten los dos formatos siguientes:

```
[etiqueta1:]  SXX  destino,1
[etiqueta2:]  SXX  destino,CL
```

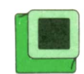
— «destino» representa el registro (que no sea de segmento) o la posición de memoria sobre la que se desea efectuar el desplazamiento.

— El segundo operando puede ser 1 o CL, según se desee desplazar un bit o más de uno. En este caso el número de bits a desplazar debe haberse cargado previamente en el registro CL.

— SXX representa cuatro posibles nemotécnicos de tres instrucciones diferentes: SHL («Shift Logical Left»), SAL («Shift Aritmetic Left», SHR («Shift Logical Right») y SAR («Shift Aritmetic Right»).

SHL y SAL realizan la misma operación, que consiste en desplazar los bits hacia el lado izquierdo completando con ceros en lado derecho. SHR desplaza los bits hacia la derecha completando con ceros en la izquierda. Por último, SAR desplaza los bits hacia la derecha repitiendo en la izquierda el bit contenido en el signo.

En estas instrucciones los flags PF, SF y ZF se actualizan de acuerdo con su significado. El flag CF contiene el último bit eliminado. Por último, en las operaciones donde se desplaza más de un bit el flag OF queda indefinido, mientras que cuando el desplazamiento es de un solo bit el flag OF queda activado cuando el bit de signo cambia de valor y desactivado en caso contrario.



Obtención de direcciones

Existen tres instrucciones que permiten obtener la dirección de una posición de memoria. Se diferencian claramente de la instrucción MOV, que transfiere el contenido de dicha posición de memoria. Estas instrucciones no afectan a los flags, y tienen el siguiente formato:

```
[etiqueta1:]  LEA  destino,origen
[etiqueta2:]  LDS  destino,origen
[etiqueta3:]  LES  destino,origen
```

— «destino» es el nombre de un registro de una palabra sobre el que se quiere cargar el «offset» de la posición de memoria deseada.

— «origen» representa la posición de memoria cuya dirección se quiere obtener.

— LEA, abreviatura de «load effective address», carga únicamente el «offset» de la posición de memoria en el registro especificado.

— LDS abreviatura de «load pointer using DS», que además del «offset» sobre el registro especificado, carga el «registro de segmento» de la posición de memoria sobre el registro DS.

— LES, abreviatura de «load pointer using ES», es análoga a la instrucción LDS, pero cargando el «registro de segmento» sobre ES.



Cadenas de información

El 8088 dispone de un mecanismo especial que permite la eficaz manipulación de largas cadenas de bytes o palabras, que de una forma genérica deno-

minaremos aquí «elementos de información» o simplemente «elementos». Dicho mecanismo basa su eficacia en que efectúa las operaciones repitiendo bucles internos al microprocesador en vez de realizar bucles de programa. Las cinco operaciones que realizan este tipo de bucles copian o comparan «elementos de información» de dos áreas, «área origen» y «área destino», o entre éstas y el acumulador.

Vamos a describir a continuación los registros que intervienen en estos bucles y los papeles que desempañan:

— El área origen es la zona de memoria que esté apuntada por los registros DS y SI. Cada vez que en un bucle se utiliza un elemento de ella, el 8088 modifica el registro SI en una o dos unidades dependiendo de si se ha utilizado un byte o una palabra.

— El área destino es la zona de memoria que esté apuntada por los registros ES y DI. Cada vez que en un bucle se utiliza un elemento de ella, el 8088 modifica el registro DI en una o dos unidades, dependiendo de si se ha utilizado un byte o una palabra.

— El flag DF («Direction Flag») se utiliza para definir si los sucesivos elementos de las áreas origen y destino se encuentran hacia las direcciones crecientes o decrecientes de la memoria. Si el flag DF está desactivado (valor 0), las modificaciones que se hacen en cada bucle de los registros SI y DI representan incrementos, y si DF está activado (valor 1), representan decrementos.

— El acumulador es el registro AL cuando se manipulan bytes o el registro AX cuando se manipulan palabras.

— Cada vez que ejecuta un bucle, el 8088 decrementa el registro CX y lo da por terminado cuando dicho registro alcanza el valor cero. Por tanto, el número de «elementos» a procesar debe definirse en CX antes de comenzar el bucle.

Los formatos de estas instrucciones son los siguientes:

```
[etiqueta1:]  [REPxx]  MOVSz
[etiqueta2:]  [REPxx]  CMPSz
[etiqueta3:]  [REPxx]  SCASz
[etiqueta4:]  [REPxx]  LODSz
[etiqueta5:]  [REPxx]  STOSz
```

Se puede observar que estas instrucciones son un poco especiales, porque no tienen operandos y porque el nemotécnico de la instrucción va precedido del prefijo de repetición (REPxx), que puede ser REP, REPE, REPZ, REPNE y REPNZ. Para las instrucciones MOVSz, LODSz y STOSz todos los prefijos tienen el mismo significado: repetir el bucle mientras CX sea distinto de cero. Para las instrucciones CMPSz y SCASz hay dos tipos de prefijos que permiten controlar el bucle de diferente forma:

— REP, REPE y REPZ significan: repetir el bucle mientras CX sea distinto de cero y mientras los «elementos» comparados sean iguales.

— REPNE y REPNZ significan: repetir el bucle mientras CX sea distinto de cero y mientras los «elementos» comparados sean diferentes.

Las cinco operaciones que se realizan dentro de los bucles pueden manipular «elementos de información» de dos tipos: bytes o palabras. Cuando se desea manejar bytes, la letra «z» final de los nemotécnicos debe ser una «B», es decir, MOVSB, CMPSB, SCASB, LODSB y STOSB y cuando se desea manejar palabras (words), la letra «z» debe ser una «W», es decir, MOVSW, CMPSW, SCASW, LODSW y STOSW.

— MOVSz copia el «elemento» del área origen al área destino.

— CMPSz resta el «elemento» del área destino del «elemento» del área origen y, sin guardar el resultado en ningún sitio, actualiza los flags permitiendo la terminación del bucle cuando se deje de cumplir la condición expresada en el prefijo.

— SCASz resta un «elemento» del área destino del acumulador y, sin guardar el resultado en ningún sitio, actualiza los flags permitiendo la terminación del bucle cuando se deje de cumplir la condición expresada en el prefijo.

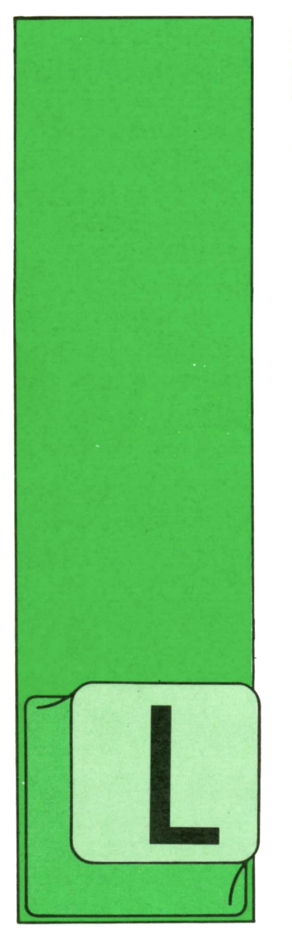
— LODSz copia el «elemento» del área origen al acumulador.

— STOSz copia el «elemento» del acumulador al área destino.

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS

Programa: representación de gráficas para IBM



OS estudiantes, sobre todo los que estudian COU, suelen necesitar de un programa como el siguiente para representación de funciones en el ordenador. Este programa nos ayudará a comprobar si la pregunta del examen sobre gráficas la hemos contestado bien. Pero no sólo sirve para esto. Muchas veces es agradable jugar con las matemáticas y ver que no sólo son números y letras. A veces es necesario recordar que muchas funciones matemáticas tienen unas representaciones muy bonitas.

El programa está organizado de una forma muy sencilla. Al principio nos aparecerá un menú. Si elegimos la opción 2 podremos definir la función que queremos representar.

Pulsando la opción 3 podremos variar algunas de las características con que se realizará la impresión, tales como la cuadrícula de la gráfica, dibujo del eje de coordenadas, dibujo de las asíntotas en caso de necesidad, dibujo por puntos o por líneas, etc.

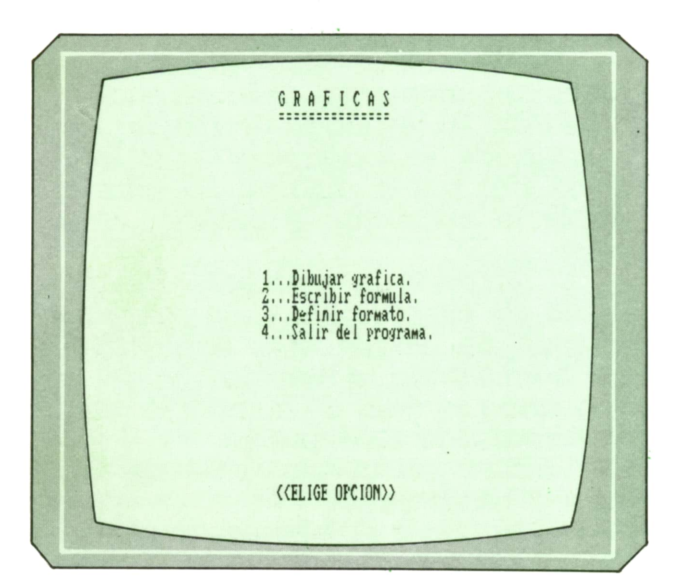




Fig. 1. Menú principal.

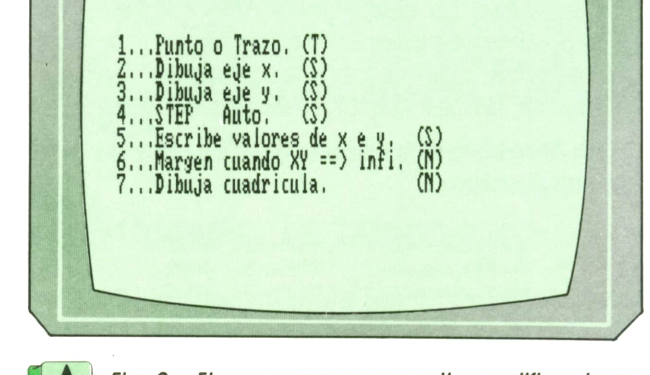




Fig. 2. El programa nos permite modificar la representación de la función.

```
PROGRAMA: FUNCIONES GRAFICAS PARA EL IBM
========================================

1000 REM ********************************************
1010 REM *                                          *
1020 REM *   F U N C I O N E S   G R A F I C A S    *
1030 REM *  ------------------------------------   *
1040 REM *                                          *
1050 REM ********************************************
1060 REM
1070 REM ********************************
1080 REM *                              *
1090 REM * (c) Ediciones Siglo Cultural *
1100 REM *                              *
1110 REM * (c) 1987                     *
1120 REM *                              *
1130 REM ********************************
1140 REM
1150 SCREEN 2
1160 REM
1170 REM **********************
1180 REM *** INICIALIZACION ***
1190 REM **********************
1200 REM
1210 OPTION BASE 1
1220 DIM A(3)
1230 LET A1=1
1240 LET A2=1
1250 LET X5=0
1260 LET AU=1
1270 DIM AU(2)
1280 DIM K1$(7)
1290 LET K1$(1)="T"
1300 LET K1$(2)="S"
1310 LET K1$(3)="S"
1320 LET K1$(4)="S"
1330 LET K1$(5)="S"
1340 LET K1$(6)="N"
1350 LET K1$(7)="N"
1360 KEY OFF
1370 FOR I=1 TO 10
1380    KEY I,""
1390 NEXT I
1400 KEY 1,CHR$(13)+"GOTO 5700"+CHR$(13)
1410 REM
1420 REM *********************************
1430 REM *** PROGRAMA PRINCIPAL "MENU" ***
1440 REM *********************************
1450 REM
1460 VIEW
1470 CLS
1480 LOCATE 2,32
1490 PRINT "G R A F I C A S"
1500 PRINT TAB(32);"==============="
1510 PRINT:PRINT:PRINT
1520 DEF FN Y(X)=(X^2+SIN(X)*X)/(X-1)
1530 PRINT:PRINT:PRINT:PRINT:PRINT
1540 PRINT SPC(29);"1...Dibujar grafica."
1550 PRINT SPC(29);"2...Escribir formula."
1560 PRINT SPC(29);"3...Definir formato."
1570 PRINT SPC(29);"4...Salir del programa."
1580 PRINT:PRINT:PRINT
1590 PRINT:PRINT:PRINT
1600 LOCATE 24,31
```

```
1610 PRINT " <<ELIGE OPCION>>";
1620 LET K$=INKEY$
1630 IF K$="" THEN GOTO 1620
1640 LET K=ASC(K$)-48
1650 IF K<1 OR K>4 THEN GOTO 1620
1660 ON K GOSUB 2070,1980,1710,1680
1670 GOTO 1460
1680 CLS
1690 END
1700 REM
1710 REM *****************************
1720 REM *** SELECCION DEL FORMATO ***
1730 REM *****************************
1740 REM
1750 CLS
1760 PRINT
1770 PRINT "  D E F I N I C I O N   D E L   F O R M A T O"
1780 PRINT "================================================"
1790 PRINT
1800 PRINT
1810 RESTORE
1820 FOR F=1 TO 7
1830    READ AF$
1840    READ NE$
1850    READ ME$
1860    PRINT ME$;
1870    IF F=6 THEN PRINT "Margen cuando XY ==> infi. (";
1880    PRINT K1$(F);")";CHR$(29);CHR$(29);
1890    LET K$=INKEY$
1900    IF K$="" THEN GOTO 1890
1910    IF ASC(K$)>90 THEN LET K$=CHR$(ASC(K$)-32)
1920    IF K$<>"" THEN IF K$=AF$ OR K$=NE$ THEN LET K1$(F)=K$
1930    IF K$<>AF$ AND K$<>NE$ THEN PRINT ELSE PRINT K1$(F)
1940 NEXT F
1950 RETURN
1960 REM
1970 REM ****************************
1980 REM *** CAMBIO DE LA FUNCION ***
1990 REM ****************************
2000 REM
2010 CLS
2020 LOCATE 22,21
2030 PRINT "Cambiar la ecuacion y pulsar F1";
2040 LOCATE 10,20
2050 EDIT 1520
2060 GOTO 1460
2070 CLS
2080 REM
2090 REM *******************
2100 REM *** DIBUJA EJES ***
2110 REM *******************
2120 REM
2130 REM *** DIBUJO DEL RECUADRO ***
2140 REM
2150 VIEW(240,0)-(639,199)
2160 WINDOW(-200,-200)-(200,200)
2170 LINE(-200,0)-(-200,200),1
2180 LINE(-200,200)-(200,200),1
2190 LINE(200,200)-(200,-200),1
2200 LINE(-200,0)-(-200,-200),1
2210 LINE(-200,-200)-(200,-200),1
2220 REM
2230 REM *** DIBUJO DEL EJE X ***
2240 REM
```

```
2250 IF K1$(2)="N" THEN GOTO 2370
2260 LINE(5,38)-(-5,38),1
2270 LINE(5,76)-(-5,76),1
2280 LINE(5,114)-(-5,114),1
2290 LINE(200,0)-(-200,0),1
2300 LINE(5,152)-(-5,152),1
2310 LINE(5,190)-(-5,190),1
2320 LINE(5,-38)-(-5,-38),1
2330 LINE(5,-76)-(-5,-76),1
2340 LINE(5,-114)-(-5,-114),1
2350 LINE(5,-152)-(-5,-152),1
2360 LINE(5,-190)-(-5,-190),1
2370 PSET(0,0)
2380 REM
2390 REM *** DIBUJO DEL EJE Y ***
2400 REM
2410 IF K1$(3)="N" THEN GOTO 2530
2420 LINE(0,200)-(0,-200),1
2430 LINE(38,5)-(38,-5),1
2440 LINE(76,5)-(76,-5),1
2450 LINE(114,5)-(114,-5),1
2460 LINE(152,5)-(152,-5),1
2470 LINE(190,5)-(190,-5),1
2480 LINE(-38,5)-(-38,-5),1
2490 LINE(-76,5)-(-76,-5),1
2500 LINE(-114,5)-(-114,-5),1
2510 LINE(-152,5)-(-152,-5),1
2520 LINE(-190,5)-(-190,-5),1
2530 IF K1$(7)="S" THEN GOSUB 5390
2540 REM
2550 REM ************************************
2560 REM *** SELECCION DE ESCALA Y AUMENTO ***
2570 REM ************************************
2580 REM
2590 LOCATE 1,1
2600 PRINT "Escala del eje X:";A1
2610 LOCATE 1,19
2620 INPUT "",A1$
2630 IF A1$<>"" THEN LET A1=VAL(A1$)
2640 LOCATE 1,19
2650 PRINT A1;"  "
2660 LOCATE 2,1
2670 PRINT "Escala del eje Y:";A2
2680 LOCATE 2,19
2690 INPUT "",A2$
2700 IF A2$<>"" THEN LET A2=VAL(A2$)
2710 LOCATE 2,19
2720 PRINT A2;"  "
2730 LOCATE 12,31
2740 PRINT "X=";INT(1000*200/A1)/1000
2750 LOCATE 1,58
2760 PRINT "Y=";INT(1000*200/A2)/1000
2770 LOCATE 3,1
2780 LET AU(1)=0
2790 PRINT "Ampliacion en eje X (0-9)";AU(1)
2800 LOCATE 3,27
2810 INPUT "",AU$(1)
2820 LOCATE 3,27
2830 PRINT AU$(1)
2840 IF AU$(1)<>"" AND AU$(1)<>"0" THEN GOTO 2850 ELSE GOTO 2900
2850 IF AU$(1)<>"1" AND AU$(1)<>"2" THEN GOTO 2860 ELSE GOTO 2900
2860 IF AU$(1)<>"3" AND AU$(1)<>"4" THEN GOTO 2870 ELSE GOTO 2900
2870 IF AU$(1)<>"5" AND AU$(1)<>"6" THEN GOTO 2880 ELSE GOTO 2900
2880 IF AU$(1)<>"7" AND AU$(1)<>"8" THEN GOTO 2890 ELSE GOTO 2900
```

```
2890 IF AU$(1)<>"9" THEN GOTO 2770
2900 LOCATE 4,1
2910 LET AU(2)=0
2920 PRINT "Ampliacion en eje Y (0-9)";AU(2)
2930 LOCATE 4,27
2940 INPUT "",AU$(2)
2950 LOCATE 4,27
2960 PRINT AU$(2)
2970 IF AU$(1)<>"" AND AU$(1)<>"0" THEN GOTO 2980 ELSE GOTO 3030
2980 IF AU$(1)<>"1" AND AU$(1)<>"2" THEN GOTO 2990 ELSE GOTO 3030
2990 IF AU$(1)<>"3" AND AU$(1)<>"4" THEN GOTO 3000 ELSE GOTO 3030
3000 IF AU$(1)<>"5" AND AU$(1)<>"6" THEN GOTO 3010 ELSE GOTO 3030
3010 IF AU$(1)<>"7" AND AU$(1)<>"8" THEN GOTO 3020 ELSE GOTO 3030
3020 IF AU$(1)<>"9" THEN GOTO 2900
3030 IF AU$(1)="" THEN AU(1)=1
3040 IF AU$(1)="0" THEN AU(1)=10
3050 IF AU$(1)="1" THEN AU(1)=20
3060 IF AU$(1)="2" THEN AU(1)=30
3070 IF AU$(1)="3" THEN AU(1)=40
3080 IF AU$(1)="4" THEN AU(1)=50
3090 IF AU$(1)="5" THEN AU(1)=60
3100 IF AU$(1)="6" THEN AU(1)=70
3110 IF AU$(1)="7" THEN AU(1)=80
3120 IF AU$(1)="8" THEN AU(1)=90
3130 IF AU$(1)="9" THEN AU(1)=100
3140 IF AU$(2)="" THEN AU(2)=1
3150 IF AU$(2)="0" THEN AU(2)=10
3160 IF AU$(2)="1" THEN AU(2)=20
3170 IF AU$(2)="2" THEN AU(2)=30
3180 IF AU$(2)="3" THEN AU(2)=40
3190 IF AU$(2)="4" THEN AU(2)=50
3200 IF AU$(2)="5" THEN AU(2)=60
3210 IF AU$(2)="6" THEN AU(2)=70
3220 IF AU$(2)="7" THEN AU(2)=80
3230 IF AU$(2)="8" THEN AU(2)=90
3240 IF AU$(2)="9" THEN AU(2)=100
3250 IF K1$(4)="S" THEN GOTO 3350
3260 REM
3270 REM ********************************
3280 REM *** SELECCION DE PASO (STEP) ***
3290 REM ********************************
3300 REM
3310 LOCATE 5,1
3320 PRINT "STEP:";S
3330 LOCATE 5,7
3340 INPUT "",S
3350 IF K1$(4)="S" THEN LET S=2/A1:LOCATE 5,1:PRINT "STEP: Autom.";S
3360 IF S=0 THEN LET S=2/A1
3370 IF K1$(1)="T" THEN GOTO 4430
3380 GOTO 3410
3390 END
3400 REM
3410 REM ********************************************
3420 REM *** TRATAMIENTO GRAFICO DE LAS FUNCIONES ***
3430 REM ********************************************
3440 REM
3450 REM **************************
3460 REM *** DIBUJAR CON PUNTOS ***
3470 REM **************************
3480 REM
3490 REM *** DIBUJO DE F(X) ***
3500 REM
3510 ON ERROR GOTO 5340
3520 LOCATE 8,1
```

```
3530 PRINT "Funcion: f(x)"
3540 PRESET(X5,0)
3550 LET XPOS=X5
3560 LET YPOS=0
3570 FOR X=X5 TO 200/A1 STEP S
3580    IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"        ":LOCATE 7,1:PRINT
 "Y:";YPOS;"        "
3590    IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 3670
3600    IF K1$(6)="N" THEN IF XPOS>205 OR XPOS<-205 OR YPOS>205 OR YPOS<-205 THE
N GOTO 3710
3610    LET PX=INT(X*A1*AU(1))
3620    LET XPOS=PX
3630    LET PY=0
3640    LET PY=INT(FN Y(X)*A2*AU(2))
3650    LET YPOS=PY
3660    PSET(PX,PY)
3670 NEXT X
3680 REM
3690 REM *** DIBUJO DE F(-X) ***
3700 REM
3710 LOCATE 8,1
3720 PRINT "funcion:f(-x)"
3730 PRINT SPACE$(28)
3740 PRESET(-200/A1,0)
3750 LET XPOS=-200/A1
3760 LET YPOS=0
3770 FOR X=-200/A1 TO X5 STEP S
3780    IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"        ":LOCATE 7,1:PRINT
 "Y:";YPOS;"        "
3790    IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 3870
3800    IF K1$(6)="N" THEN IF XPOS>205 OR XPOS<-205 OR YPOS>205 OR YPOS<-205 THE
N GOTO 3910
3810    LET PX=INT(X*A1*AU(1))
3820    LET XPOS=PX
3830    LET PY=0
3840    LET PY=INT(FN Y(X)*A2*AU(2))
3850    LET YPOS=PY
3860    PSET(PX,PY)
3870 NEXT X
3880 REM
3890 REM *** DIBUJO DE -F(X) ***
3900 REM
3910 LOCATE 8,1
3920 PRINT "Funcion:-f(x)"
3930 PRINT SPACE$(28)
3940 PRESET(X5,0)
3950 LET XPOS=X5
3960 LET YPOS=0
3970 FOR X=X5 TO 200/A1 STEP S
3980    IF K1$(5)="5" THEN LOCATE 6,1:PRINT "X:";XPOS;"        ":LOCATE 7,1:PRINT
 "Y:";YPOS;"        "
3990    IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 4070
4000    IF K1$(6)="N" THEN IF XPOS>205 OR YPOS>205 OR XPOS<-205 OR YPOS<-205 THE
N GOTO 4110
4010    LET PX=INT(X*A1*AU(1))
4020    LET XPOS=PX
4030    LET PY=0
4040    LET PY=INT(-(FN Y(X)*A2*AU(2)))
4050    LET YPOS=PY
4060    PSET(PX,PY)
4070 NEXT X
4080 REM
4090 REM *** DIBUJO DE -F(-X) ***
4100 REM
```

```
4110 LOCATE 8,1
4120 PRINT "Funcion:-f(-x)"
4130 PRINT SPACE$(28)
4140 PRESET(-200/A1,0)
4150 LET XPOS=-200/A1
4160 LET YPOS=0
4170 FOR X=-200/A1 TO X5 STEP S
4180     IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"        ":LOCATE 7,1:PRINT
 "Y:";YPOS;"        "
4190     IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 4270
4200     IF K1$(6)="N" THEN IF XPOS>205 OR YPOS>205 OR XPOS<-205 OR YPOS<-205 THE
N GOTO 4280
4210     LET PX=INT(X*A1*AU(1))
4220     LET XPOS=PX
4230     LET PY=0
4240     LET PY=INT(-(FN Y(X)*A2*AU(2)))
4250     LET YPOS=PY
4260     PSET(PX,PY)
4270 NEXT X
4280 LOCATE 12,1
4290 PRINT "*****************"
4300 LOCATE 13,1
4310 PRINT "*PULSA UNA TECLA*"
4320 LOCATE 14,1
4330 PRINT "*****************"
4340 LET A$=INPUT$(1)
4350 GOTO 1460
4360 REM
4370 REM *************************
4380 REM *** DIBUJAR CON TRAZO ***
4390 REM *************************
4400 REM
4410 REM *** DIBUJO DE F(X) ***
4420 REM
4430 ON ERROR GOTO 5340
4440 LOCATE 8,1
4450 PRINT "Funcion: f(x)"
4460 PRINT SPACE$(28)
4470 PRESET(X5,0)
4480 LET XPOS=X5
4490 LET YPOS=0
4500 FOR X=X5 TO 200/A1 STEP S
4510     IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"        ":LOCATE 7,1:PRINT
 "Y:";YPOS;"        "
4520     IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 4600
4530     IF K1$(6)="N" THEN IF XPOS>205 OR XPOS<-205 OR YPOS>205 OR YPOS<-205 THE
N GOTO 4640
4540     LET PX=INT(X*A1*AU(1))
4550     LET XPOS=PX
4560     LET PY=0
4570     LET PY=INT(FN Y(X)*A2*AU(2))
4580     LET YPOS=PY
4590     LINE -(PX,PY),1
4600 NEXT X
4610 REM
4620 REM *** DIBUJO DE F(-X) ***
4630 REM
4640 LOCATE 8,1
4650 PRINT "funcion:f(-x)"
4660 PRINT SPACE$(28)
4670 PRESET(-200,0)
4680 LET XPOS=-200
4690 LET YPOS=0
4700 FOR X=-200/A1 TO X5 STEP S
```

```
4710    IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"         ":LOCATE 7,1:PRINT
 "Y:";YPOS;"         "
4720    IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 4800
4730    IF K1$(6)="N" THEN IF XPOS>205 OR XPOS<-205 OR YPOS>205 OR YPOS<-205 THE
N GOTO 4810
4740    LET PX=INT(X*A1*AU(1))
4750    LET XPOS=PX
4760    LET PY=0
4770    LET PY=INT(FN Y(X)*A2*AU(2))
4780    LET YPOS=PY
4790    LINE -(PX,PY),1
4800 NEXT X
4810 LOCATE 8,1
4820 REM
4830 REM *** DIBUJO DE -F(X) ***
4840 REM
4850 PRINT "funcion:-f(x)"
4860 PRINT SPACE$(28)
4870 PRESET(X5,0)
4880 LET XPOS=X5
4890 LET YPOS=0
4900 FOR X=X5 TO 200/A1 STEP S
4910    IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"         ":LOCATE 7,1:PRINT
 "Y:";YPOS;"         "
4920    IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 5000
4930    IF K1$(6)="N" THEN IF XPOS>205 OR XPOS<-205 OR YPOS>205 OR YPOS<-205 THE
N GOTO 5040
4940    LET PX=INT(X*A1*AU(1))
4950    LET XPOS=PX
4960    LET PY=0
4970    LET PY=INT(-(FN Y(X)*A2*AU(2)))
4980    LET YPOS=PY
4990    LINE -(PX,PY),1
5000 NEXT X
5010 REM
5020 REM *** DIBUJO DE -F(-X) ***
5030 REM
5040 LOCATE 8,1
5050 PRINT "funcion:-f(-x)"
5060 PRINT SPACE$(28)
5070 PRESET(-200,0)
5080 LET XPOS=-200
5090 LET YPOS=0
5100 FOR X=-200/A1 TO X5 STEP S
5110    IF K1$(5)="S" THEN LOCATE 6,1:PRINT "X:";XPOS;"         ":LOCATE 7,1:PRINT
 "Y:";YPOS;"         "
5120    IF X*A1>32750 OR Y*A2>32750 OR X*A1<-32750 OR Y*A2<-32750 THEN GOTO 5200
5130    IF K1$(6)="N" THEN IF XPOS>205 OR XPOS<-205 OR YPOS>205 OR YPOS<-205 THE
N GOTO 5210
5140    LET PX=INT(X*A1*AU(1))
5150    LET XPOS=PX
5160    LET PY=0
5170    LET PY=INT(-(FNY(X)*A2*AU(2)))
5180    LET YPOS=PY
5190    LINE -(PX,PY),1
5200 NEXT X
5210 LOCATE 12,1
5220 PRINT "*****************"
5230 LOCATE 13,1
5240 PRINT "*PULSA UNA TECLA*"
5250 LOCATE 14,1
5260 PRINT "*****************"
5270 LET A$=INPUT$(1)
5280 GOTO 1460
```

```
5290 REM
5300 REM ******************************
5310 REM *** TRATAMIENTO DE ERRORES ***
5320 REM ******************************
5330 REM
5340 LOCATE 9,1
5350 PRINT "** ERROR. SOBREPASA RANGO **"
5360 RESUME NEXT
5370 END
5380 REM
5390 REM *************************
5400 REM *** DUBUJA CUADRICULA ***
5410 REM *************************
5420 REM
5430 LINE(200,38)-(-200,38),1
5440 LINE(200,76)-(-200,76),1
5450 LINE(200,114)-(-200,114),1
5460 LINE(200,152)-(-200,152),1
5470 LINE(200,190)-(-200,190),1
5480 LINE(200,-38)-(-200,-38),1
5490 LINE(200,-76)-(-200,-76),1
5500 LINE(200,-114)-(-200,-114),1
5510 LINE(200,-152)-(-200,-152),1
5520 LINE(200,-190)-(-200,-190),1
5530 LINE(38,200)-(38,-200),1
5540 LINE(76,200)-(76,-200),1
5550 LINE(114,200)-(114,-200),1
5560 LINE(152,200)-(152,-200),1
5570 LINE(190,200)-(190,-200),1
5580 LINE(-38,200)-(-38,-200),1
5590 LINE(-76,200)-(-76,-200),1
5600 LINE(-114,200)-(-114,-200),1
5610 LINE(-152,200)-(-152,-200),1
5620 LINE(-190,200)-(-190,-200),1
5630 RETURN
5640 REM
5650 REM ******************
5660 REM *** LINEAS DATA ***
5670 REM ******************
5680 REM
5690 DATA "P","T","1...Punto o Trazo.  ("
5700 DATA "S","N","2...Dibuja eje x.   ("
5710 DATA "S","N","3...Dibuja eje y.   ("
5720 DATA "S","N","4...STEP    Auto.    ("
5730 DATA "S","N","5...Escribe valores de x e y.   ("
5740 DATA "S","N","6..."
5750 DATA "S","N","7...Dibuja cuadricula.          ("
5760 REM
5770 REM ******************************************
5780 REM *** FIN DE LA DEFINICION DE LA FUNCION ***
5790 REM ******************************************
5800 REM
5810 VIEW
5820 CLS
5830 PRINT "Si has definido el formato, tendras que volver a hacerlo."
5840 PRINT
5850 PRINT "PULSA UNA TECLA"
5860 LET A$=INPUT$(1)
5870 RUN
```

El programa está preparado para dibujar la función y su inversa. Lo que realmente hace el programa es representar:

f (x)
f (–x)
–f (x)
–f (–x)

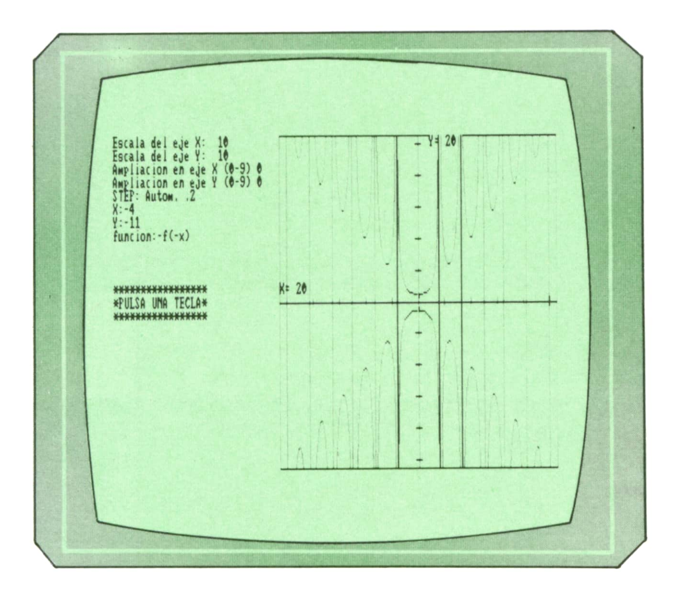




Fig. 3. Representación de y = sen (x) + x.

Una vez que hayamos elegido la opción 1 para representar la función, el programa nos pedirá algunos datos como la escala en X y en Y de los ejes donde se visualizará la función.

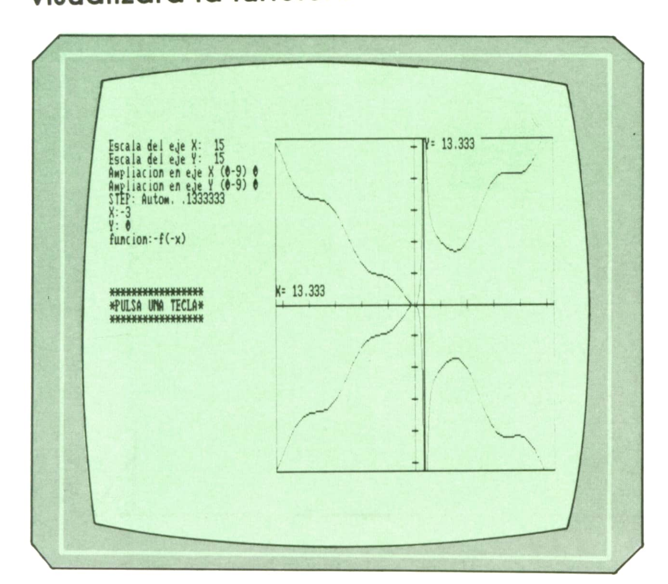




Fig. 4. Representación de y = sen (x) /x.

También nos preguntará si queremos ampliación en X, en Y o en ambas a la vez. Con esto podemos hacer que una parte de la función se vea mejor y más grande.

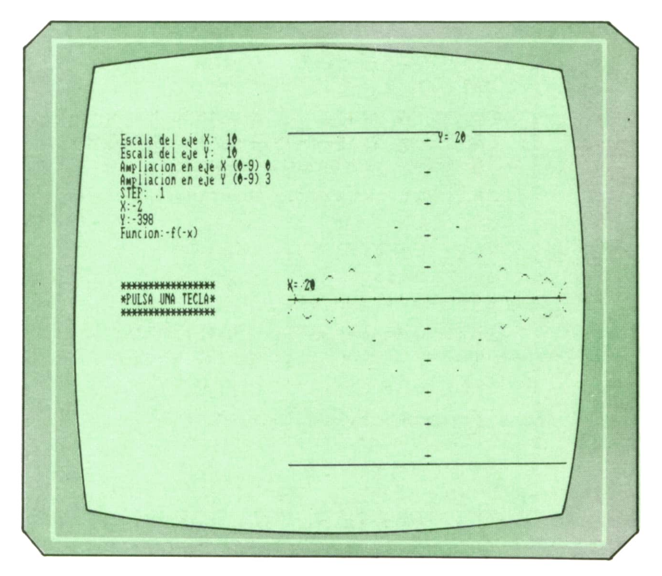




Fig. 5. Representación de y = .

Aunque al principio los resultados no sean los deseados, no hay que preocuparse. Es necesario jugar un poco con la graduación del eje de coordenadas y con la amplitud en X y en Y para obtener un mayor rendimiento del programa.



Programa: ácidos e hidróxidos para Spectrum

Aparece a continuación el segundo programa realizado específicamente para SPECTRUM con el que podremos repasar nuestros conocimientos sobre química. En este caso veremos la formulación de los **ACIDOS** (hidrácidos y oxácidos) e **HIDROXIDOS.**

El programa nos preguntará nuestro nombre e, inmediatamente, comenzará a preguntarnos. En la pantalla nos aparecerán una serie de símbolos químicos y el nombre de uno de ellos. Tenemos que pulsar el número que se encuentra junto a la respuesta correcta.

En caso de no acertar, el ordenador nos indicará cuál es la respuesta correc-

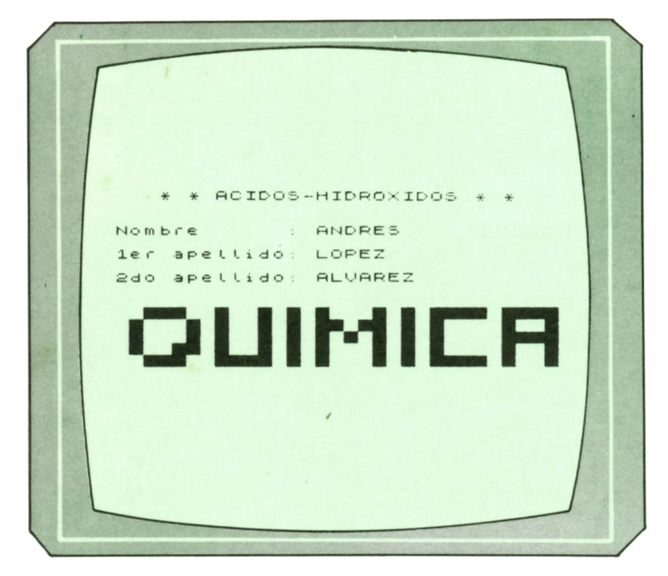



Fig. 6. Presentación del programa.

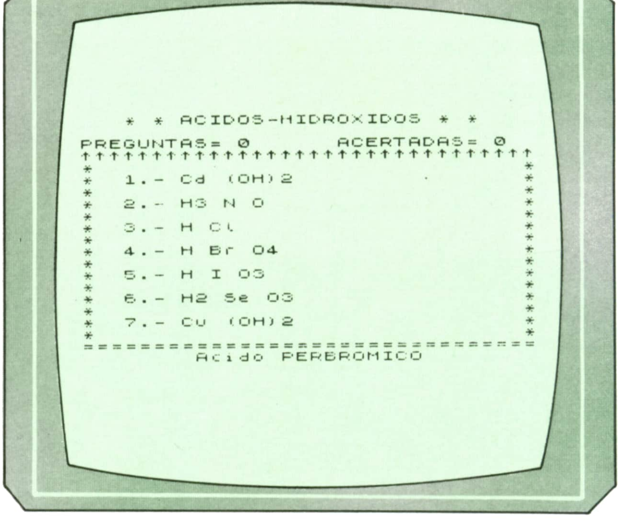



Fig. 7.

ta. En cualquier caso, tras contestar una pregunta le podemos decir al ordenador que queremos terminar el test pulsando la letra «A». Cuando lo hagamos nos aparecerá en pantalla una pequeña estadística que nos contará cómo lo hemos hecho.

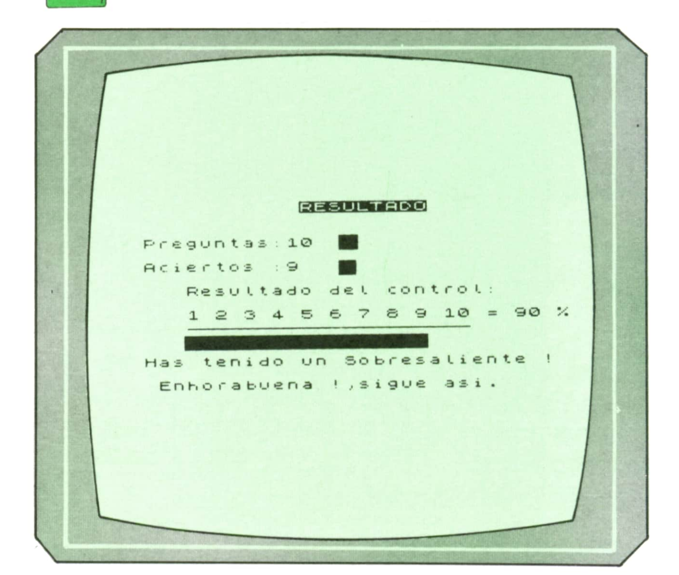


Fig. 8. Estadística final.



```
PROGRAMA: ACIDOS Y HIDROXIDOS
------------------------------

  10 REM ***********************
  20 REM ***********************
  30 REM ******* QUIMICA *******
  40 REM ***********************
  50 REM ** POR CARLOS DORAL ***
  60 REM ***********************
  70 REM ***********************
  80 REM ******* ACIDOS ********
  90 REM ***********************
 100 REM ***** HIDROXIDOS ******
 110 REM ***********************
 120 REM ***********************
 125 REM
 130 DIM r$(7,8)
 140 DIM p(87)
 150 CLS
 160 PRINT AT 0,3; FLASH 1;"* * ACIDOS-HIDROXIDOS * *"
 170 PRINT AT 3,0;"Nombre        : "
```

```
180 PRINT AT 5,0;"1er apellido: "
190 PRINT AT 7,0;"2do apellido: "
200 POKE 23658,8
210 RESTORE 260
220 FOR f=10 TO 14
230    READ a$
240    PRINT AT f,1;a$
250 NEXT f
260 DATA " ***   *   * * *   * * **** ****"
270 DATA "*    * *   * * ** ** * *     *  *"
280 DATA "*    * *   * * * * * * *     ****"
290 DATA "*   ** *   * * *   * * *     *  *"
300 DATA " ***   **** * *   * * **** *  *"
310 INPUT "Nombre: "; LINE n$
320 IF LEN n$<1 OR LEN n$>18 THEN  GO TO 310
330 PRINT AT 3,14; PAPER 4;n$
340 INPUT "1er apellido: "; LINE a$
350 IF LEN a$<1 OR LEN a$>18 THEN  GO TO 340
360 PRINT AT 5,14; PAPER 6;a$
370 INPUT "2 apellido: "; LINE b$
380 IF LEN b$<1 OR LEN b$>18 THEN  GO TO 370
390 PRINT AT 7,14; PAPER 6;b$
400 PRINT  INK 4;#0; FLASH 1;"         PULSA UNA TECLA
410 FOR c=0 TO 6
420    FOR f=10 TO 14
430      PRINT  INK c;AT f,0; OVER 1;"
440    NEXT f
450    IF INKEY$<>"" THEN  GO TO 480
460 NEXT c
470 GO_TO 410
480 LET b=0
490 LET p=0
500 CLS
510 PRINT AT 2,0;"PREGUNTAS= ";p;"        ACERTADAS= ";b
520 RANDOMIZE 0
530 DIM r(87)
540 PRINT AT 0,3; FLASH 1;"* * ACIDOS-HIDROXIDOS * *"
550 PRINT AT 3,0;"^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"
560 PRINT AT 19,0;"================================"
570 FOR f=4 TO 18
580    PRINT AT f,0;"*";AT f,31;"*"
590 NEXT f
600 LET rn=1+INT (RND*87)
610 IF p(rn)=1 THEN  GO TO 600
620 LET p(rn)=1
630 LET e=1670+(rn*10)
640 RESTORE e
650 READ p$
660 READ o$
670 IF rn>44 AND rn<52 THEN  LET a$="Acido "+p$+"HIDRICO"
680 IF rn<44 THEN  LET a$="Acido "+p$
690 IF rn>52 THEN  LET a$="Hidroxido "+p$
700 FOR f=1 TO 7
710    LET a=INT (RND*86)+1
720    IF r(a)=1 THEN  GO TO 710
730    LET r(a)=1
740    LET e=(1670+(10*a))
750    RESTORE e
760    READ r$(f)
770    READ r$(f)
780 NEXT f
790 LET ac=(INT (RND*6)+1)
800 LET r$(ac)=o$
810 LET y=1
820 FOR f=5 TO 17 STEP 2
830    PRINT AT F,3;y;".- ";r$(y)
```

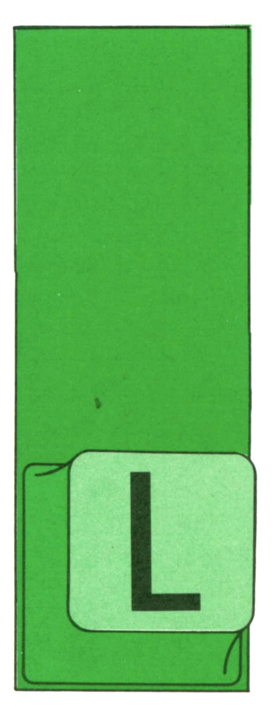
```
 840    LET y=y+1
 850 NEXT f
 860 PRINT AT 20,(32-LEN a$)/2;a$
 870 LET k$=INKEY$
 880 IF CODE k$<49 OR CODE k$>55 THEN  GO TO 870
 890 IF VAL k$=ac THEN  GO SUB 1420: GO TO 910
 900 GO SUB 1510
 910 LET p=p+1
 920 PRINT #0;"      A-ACABAR       S-SEGUIR      "
 930 LET k$=INKEY$
 940    IF k$="A" OR k$="a" THEN  GO TO 970
 950    IF k$="S" OR k$="s" THEN  GO TO 500
 960 GO TO 930
 970 REM REPRESENTACION GRAFICA
 980 CLS
 990 PRINT AT 0,11; INVERSE 1;"RESULTADO"
1000 PRINT AT 3,0;"Preguntas:";p
1010 PRINT AT 5,0;"Aciertos :";b
1020 FOR f=110 TO 110+p
1030 PLOT  INK 1;f,152
1040 DRAW  INK 1;0,-9
1050 NEXT f
1060 IF b=0 THEN  GO TO 1110
1070 FOR f=110 TO 110+b
1080 PLOT  INK 3;f,135
1090 DRAW  INK 3;0,-9
1100 NEXT f
1110 PRINT AT 7,3;"Resultado del control:"
1120 PLOT 25,90
1130 DRAW 158,0
1140 PRINT AT 9,3;"1 2 3 4 5 6 7 8 9 10"
1150 LET x=23
1160 LET y=85
1170 LET t=((b/p)*100)
1180 PRINT AT 9,24;"= ";INT t;" % "
1190 LET r=INT (t/10)
1200 LET r=r*10
1210 LET r=t-r
1220 LET t=t/10
1230 FOR f=x TO ((x+INT t*16)-8)+r
1240 PLOT  INK 4;f,y
1250 DRAW  INK 4;0,-9
1260 NEXT f
1270 LET i$="Muy deficiente"
1280 LET t$="No has tocado ni un libro,no te da verguenza ?"
1290 IF t>2 AND t<5 THEN  LET i$="Insuficiente": LET t$="Fuera de mi vista !,has
ta que noestudies mas no jugaras conmigo."
1300 IF t>4 AND t<6 THEN  LET i$="Suficiente": LET t$="Tienes que estudiar mas."
1310 IF t>5 AND t<7 THEN  LET i$="Bien": LET t$="Muy bien.": GO TO 1340
1320 IF t>6 AND t<9 THEN  LET i$="Notable": LET t$="Estupendo.Vas bastante bien
Don "+n$
1330 IF t>8 THEN  LET i$="Sobresaliente !": LET t$=" Enhorabuena !,sigue asi."
1340 PRINT AT 15,0;t$
1350 PRINT AT 13,0;"Has tenido un "; FLASH 1;i$
1360 INPUT "Quieres repetir (S/N) ?";t$
1370 IF t$="S" OR t$="s" THEN  GO TO 150
1380 STOP
1390 CLEAR
1400 SAVE "ACIDOS-HDX" LINE 130
1410 STOP
1420 REM **********************
1430 REM ** RESPUESTA ACERTADA *
1440 REM **********************
1450 PRINT AT 20,5; FLASH 1;" * * * MUY BIEN * * * "
1460 FOR f=1 TO 40
1470    BEEP .03,INT (RND*50)
1480 NEXT f
```

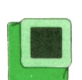
```
1490 LET b=b+1
1500 RETURN
1510 REM ***********************
1520 REM ** RESPUESTA FALLADA **
1530 REM ***********************
1540 PRINT AT 20,0; FLASH 1;"         OOOH,FALLASTE !
1550 FOR F=0 TO -15 STEP -1
1560    BEEP .10,F
1570 NEXT F
1580 FOR f=1 TO 50
1590  NEXT f
1600 PRINT AT 20,0;"     LA CORRECTA ERA  ";o$;"
1610 PRINT AT 3+(ac*2),3; FLASH 1; OVER 1;TAB 27
1620 FOR f=1 TO 200
1630 NEXT f
1640 RETURN
1650 REM ***********************
1660 REM **PREGUNTAS-RESPUESTAS*
1670 REM ***********************
1680 DATA "HIPOCLOROSO","H Cl O"
1690 DATA "CLOROSO","H Cl O2"
1700 DATA "CLORICO","H Cl O3"
1710 DATA "PERCLORICO","H Cl O4"
1720 DATA "HIPOFLUOROSO","H F O"
1730 DATA "FLUOROSO","H F O2"
1740 DATA "FLUORICO","H F O3"
1750 DATA "PERFLUORICO","H F O4"
1760 DATA "HIPOBROMOSO","H Br O"
1770 DATA "BROMOSO","H Br O2"
1780 DATA "BROMICO","H Br O3"
1790 DATA "PERBROMICO","H Br O4"
1800 DATA "HIPOYODOSO","H I O"
1810 DATA "YODOSO","H I O2"
1820 DATA "YODICO","H I O3"
1830 DATA "PERYODICO","H I O4"
1840 DATA "HIPOSULFUROSO","H2 S O2"
1850 DATA "SULFUROSO","H2 S O3"
1860 DATA "SULFURICO","H2 S O4"
1870 DATA "HIPOSELENIOSO","H2 Se O2"
1880 DATA "SELENIOSO","H2 Se O3"
1890 DATA "SELENICO","H2 Se O4"
1900 DATA "HIPOTELUROSO","H2 Te O2"
1910 DATA "TELUROSO","H2 Te O3"
1920 DATA "TELURICO","H2 Te O4"
1930 DATA "HIPONITROSO","H3 N O"
1940 DATA "NITROSO","H3 N O2"
1950 DATA "NITRICO","H3 N O3"
1960 DATA "HIPOFOSFOROSO","H3 P O"
1970 DATA "FOSFOROSO","H3 P O2"
1980 DATA "FOSFORICO","H3 P O3"
1990 DATA "HIPOARSENIOSO","H3 As O"
2000 DATA "ARSENIOSO","H3 As O2"
2010 DATA "ARSENICO","H3 As O3"
2020 DATA "HIPOANTIMONIOSO","H3 Sb O"
2030 DATA "ANTIMONIOSO","H3 Sb O2"
2040 DATA "ANTIMONICO","H3 Sb O3"
2050 DATA "HIPOBISMUTOSO","H3 Bi O"
2060 DATA "BISMUTOSO","H3 Bi O2"
2070 DATA "BISMUTICO","H3 Bi O3"
2080 DATA "CARBONOSO","H2 Co O2"
2090 DATA "CARBONICO","H2 Co O4"
2100 DATA "SILICIOSO","H2 Si O2"
2110 DATA "SILICICO","H2 Si O4"
2120 DATA "CLOR","H Cl"
2130 DATA "FLUOR","H F"
2140 DATA "BROM","H Br"
2150 DATA "YOD","H I"
```

```
2160 DATA "SULF","H2 S"
2170 DATA "SELEN","H2 Se"
2180 DATA "TELUR","H2 Te"
2190 DATA "LITICO","Li (OH)"
2200 DATA "SODICO","Na (OH)"
2210 DATA "POTASICO","K (OH)"
2220 DATA "RUBIDICO","Rb (OH)"
2230 DATA "CESICO","Cs (OH)"
2240 DATA "ARGENTIOSO","Ag (OH)"
2250 DATA "ARGENTICO","Ag (OH)2"
2260 DATA "CUPROSO","Cu (OH)"
2270 DATA "CUPRICO","Cu (OH)2"
2280 DATA "AUROSO","Au (OH)"
2290 DATA "AURICO","Au (OH)3"
2300 DATA "CALCICO","Ca (OH)2"
2310 DATA "BERILICO","Be (OH)2"
2320 DATA "ESTRONCICO","Sr (OH)2"
2330 DATA "BARICO","Ba (OH)2"
2340 DATA "RADICO","Ra (OH)2"
2350 DATA "MAGNESICO","Mg (OH)2"
2360 DATA "ZINCICO","Zn (OH)2"
2370 DATA "CADMICO","Cd (OH)2"
2380 DATA "FERROSO","Fe (OH)2"
2390 DATA "FERRICO","Fe (OH)3"
2400 DATA "NIQUELOSO","Ni (OH)2"
2410 DATA "NIQUELICO","Ni (OH)3"
2420 DATA "CROMOSO","Cr (OH)2"
2430 DATA "CROMICO","Cr (OH)3"
2440 DATA "COBALTOSO","Co (OH)2"
2450 DATA "COBALTICO","Co (OH)3"
2460 DATA "MANGANOSO","Mn (OH)2"
2470 DATA "MANGANICO","Mn (OH)3"
2480 DATA "ALUMINICO","Al (OH)3"
2490 DATA "BORICO","Bo (OH)3"
2500 DATA "PLUMBOSO","Pb (OH)2"
2510 DATA "PLUMBICO","Pb (OH)4"
2520 DATA "ESTANNOSO","Sn (OH)2"
2530 DATA "ESTANNICO","Sn (OH)4"
2540 DATA "PLATINICO","Pt (OH)4"
2550 DATA "IRIDICO","Ir (OH)4"
```

TECNICAS DE ANALISIS

PROYECTOS DE AUTOMATIZACION DE LA PRODUCCION





Análisis y concepción

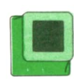
LA segunda fase del desarrollo de un proyecto de automatización industrial es la que, desde nuestro punto de vista (el del analista), resulta más interesante. En ella toman forma todas las informaciones previamente obtenidas y estudiadas, y es la tarea más creativa y comprometida.

Como resultado de esta fase deberán quedar claramente establecidos:

— Las características de los materiales a los que se refiere el proyecto en cuestión: es sumamente importante esta definición no sólo para la adecuada tipificación de los procesos y tareas que posteriomente se han de concretar, sino para futuras consultas (incluso de proveedores, en la fase de explotación del sistema).

— Las especificaciones del sistema de planificación de la producción que se propone, precisando si se va a utilizar algún software específico para ello (o, incluso, si ha de diseñarse una aplicación «ad hoc»).

— Los procedimientos de supervisión y organización del sistema de producción.



Fases del análisis

Las diferentes fases en que suele desarrollarse el análisis y concepción de un sistema como el que nos ocupa, son las que se muestran en la figura y que podríamos resumir como:

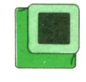
— Recopilar y analizar los datos obtenidos.

— Concebir el sistema automatizado de producción.

— Redactar los cuadernos de cargas, especificaciones y procedimientos.

— Evaluar y redactar el proyecto.

— Decidir.



Métodos de análisis y concepción de sistemas de control y supervisión

Los dos métodos más extendidos que se utilizan específicamente para los sistemas de control de la producción son el GRAI y el SADT.

Método GRAI

El método GRAI ha sido desarrollado en el laboratorio de Automática de la Universidad de Burdeos, como resultado de un proyecto (de diez años de duración) de control en tiempo real de una unidad de fabricación.

Este método se basa en la representación de la realidad mediante dos útiles gráficos (rejillas y redes) y su manejo, en la tarea de análisis, de acuerdo con unas reglas específicas.

La «rejilla GRAI» es una tabla de doble entrada donde se representan todos los elementos significativos en el proceso de producción; en la parte superior, como encabezamiento de las diferentes columnas, se reseñan las diversas *funciones* involucradas en el proceso en estudio (una *función* está formada por el conjunto de actividades que tienen una finalidad común identificable: planificar, gestionar recursos, aprovisionar, etc.); en las sucesivas filas se incluyen los elementos por niveles (se define un *nivel* como el conjunto de actividades que tienen el mismo horizonte y el mismo período; por tanto, en cada casilla aparecerá un *centro de actividades* (reunión de todas las actividades que pertenecen a la misma

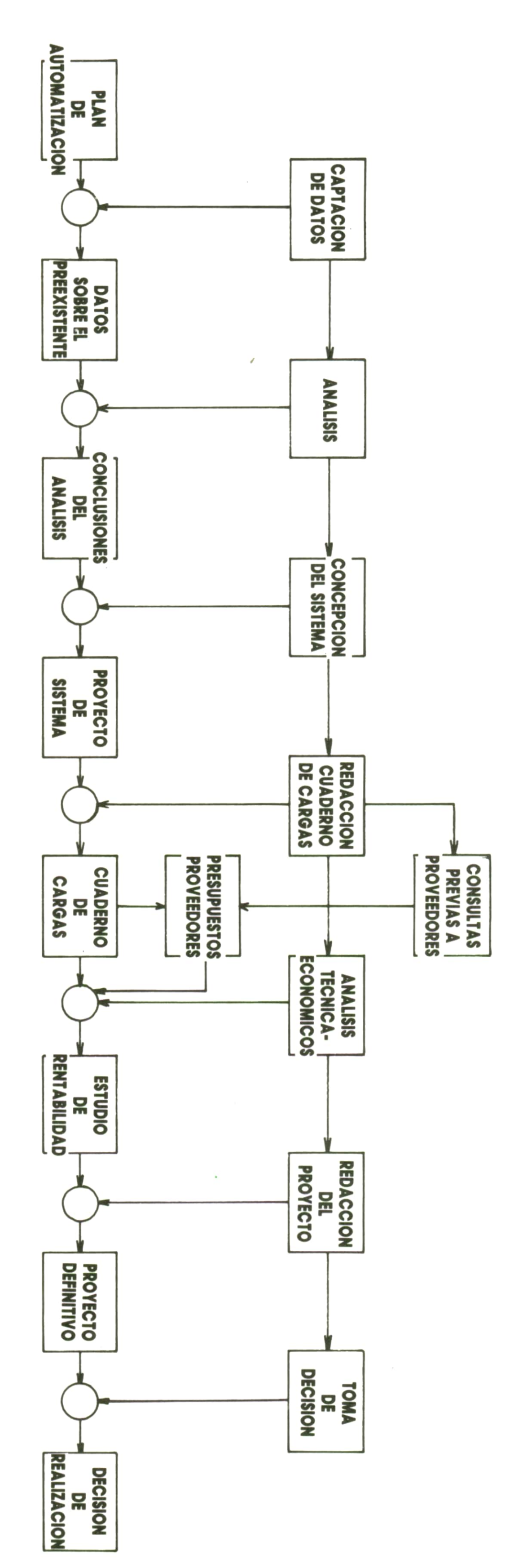



Fig. Fases del desarrollo del análisis.

función y nivel); también se utiliza el *cuadro de decisión*, formado por las informaciones disponibles acerca de los objetivos impuestos a un centro de actividades y sobre las variables susceptibles de ser modificadas respecto a las decisiones a tomar que conciernen a ese centro.

Las «redes GRAI» permiten representar las diferentes actividades de la función de gestión de la producción, poniendo de manifiesto los componentes de cada actividad y las operaciones sobre los resultados y los soportes. El elemento básico de la red es el *soporte* (objeto físico o de información que interviene en la realización de una actividad); se consideran dos tipos básicos de soportes: los *soportes-recursos* (objeto, material, herramienta, etc. que interviene en la tarea y pertenece al sistema *físico de desarrollo de la actividad) y los soportes-información* (soporte-resultado, soporte-disparador-de-una-tarea, soporte-perturbación, etcétera). Mediante una adecuada simbología (muy precisa y concreta), se representan todos los flujos de materiales e informaciones involucrados en las actividades.

Método SADT

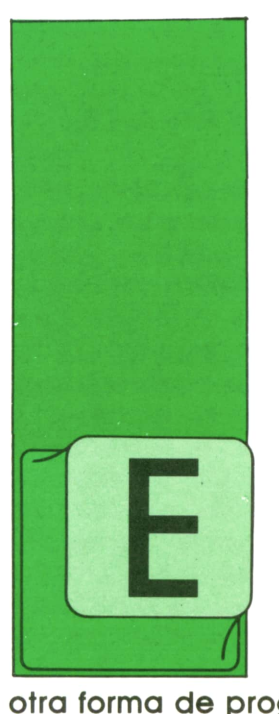
Fue desarrollado en el cuadro de un proyecto de las Fuerzas Armadas de los EE.UU. de América. El objetivo básico del método, según declaran sus creadores, es facilitar la aproximación analítica al sistema de producción.

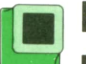
Se apoya en dos principios básicos: definir el sistema a analizar determinando: *a)* Las fronteras del sistema; *b)* El criterio de análisis, y *c)* El objetivo del análisis; y, en segundo lugar, construir una jerarquía de las actividades realizadas por el sistema y los medios necesarios.

La descripción del sistema se realiza utilizando los diagramas de actividades (llamados *actigramas)* y los diagramas de datos (*datagramas)*.

El estudio se desarrolla en dos fases: una de análisis (elaboración de diagramas, establecimiento de ligaduras entre los diagramas, indicación de secuencia de actividades, etc.) y otra de concepción de la solución adecuada. Normalmente este método se utiliza con «herramientas» informáticas que se han desarrollado al respecto.

TECNICAS DE PROGRAMACION





Programación no algorítmica

EN el capítulo anterior describí someramente algunas de las técnicas que podrían hacer posible la programación en paralelo. En esta ocasión vamos a hablar de otra forma de programar, que ha cobrado gran interés a partir de hace poco más de una década.

Todos los ejemplos presentados en los 39 capítulos anteriores han sido siempre casos particulares de la «programación procedimental» o «programación algorítmica». Recordemos (véase el capítulo primero) que se llama algoritmo a un conjunto de reglas y pasos bien definidos que permiten realizar un cálculo determinado. Pues bien, todos los programas escritos en lenguajes de programación «clásicos» (como BASIC, PASCAL, APL, etc.) no son, en el fondo, otra cosa que algoritmos. En efecto: un programa escrito en uno de esos lenguajes es un conjunto de instrucciones que se ejecutan en un orden determinado y completamente predecible, en función de los datos particulares a los que se aplica y sobre los que se trabaja. Por esta razón, los programas clásicos, como los algoritmos, pueden representarse en forma de organigramas, de lo que hemos dado abundantes pruebas a lo largo de esta sección.

En 1972 el francés A. Colmerauer inventó un nuevo lenguaje de programación que se apartaba considerablemente de todos los anteriores. En efecto, este lenguaje estaba basado en las leyes de la lógica matemática y se afirmaba que su funcionamiento debía asemejarse, más que el de los lenguajes algorítmicos, a la forma de pensar del hombre. Su nombre (PROLOG) es un acrónimo de la frase inglesa «PROgramming in LOGic» (o de la francesa «PROgrammation LOGique». El lenguaje fue mejorado, en años sucesivos, por los británicos Warren y Kowalski (Edimburgo, 1974).

En contraposición a los lenguajes de programación clásica, los programas escritos en PROLOG no son algorítmicos. Dicho de otra manera: sus instrucciones no se ejecutan en un orden predeterminado. Por esta razón, los programas PROLOG no son representables en forma de organigramas, y se dice que el lenguaje PROLOG es «no procedimental», porque en él no pueden construirse procedimientos semejantes a los de PASCAL y otros muchos lenguajes clásicos.

Con la llegada de la década de los ochenta, los japoneses lanzaron el proyecto de la «quinta generación de ordenadores». Se trataba de un proyecto muy ambicioso, que en el transcurso de una década tenía como objetivo la sustitución de las computadoras actuales por otras de nuevo diseño, basadas en las técnicas de la inteligencia artificial, que deberían ser capaces de resolver problemas tan complejos como el análisis del lenguaje natural (podríamos comunicarnos con los ordenadores en nuestra lengua nativa) y la traducción automática, lo que provocaría una nueva revolución tecnológica, que cambiaría la estructura de la sociedad tan drásticamente como las anteriores. No parece que estas previsiones vayan a cumplirse, al menos en el futuro inmediato, pero es curioso observar que el lenguaje escogido por los investigadores japoneses para servir de base a su quinta generación fue, precisamente, PROLOG.

Los datos de un programa PROLOG se definen en una forma especial, basada en el cálculo de predicados de primer orden. Recuérdese que un predicado no es otra cosa que la afirmación de una

propiedad de una o varias entidades (personas, objetos, etc.). Por ejemplo, al afirmar que «Luisa es mujer», enunciamos una propiedad (ser mujer) que se aplica, precisamente a Luisa. Con la nomenclatura de PROLOG, esta frase (este dato) se expresaría así:

mujer (luisa).

De igual manera podemos expresar una propiedad algo más compleja, que afecte a dos entidades distintas. Por ejemplo, decir que «Juan es padre de Luisa» define una propiedad (una relación) que liga a Juan con Luisa. Con la nomenclatura de PROLOG, esto se expresará así:

padre (juan, luisa)

Las instrucciones del lenguaje PROLOG tienen también una forma muy especial. En realidad, existe una forma única (llamada «cláusula de Horn») que puede escribirse de una u otra manera dependiendo del intérprete o compilador concreto con que nos encontremos. Una de estas formas podría ser la siguiente:

L1 <–L2 & L3 ... & Ln.

donde L1, L2, L3, ..., Ln no son otra cosa que predicados parecidos a los de los ejemplos anteriores, aunque es posible que una o más de las entidades de un predicado pueda sustituirse por una variable (los nombres de las variables, en PROLOG, suelen comenzar por un asterisco o una letra mayúscula, aunque esta regla no es absolutamente general y depende del intérprete o compilador). Pues bien: la instrucción anterior representa una estructura condicional que se puede leer de la siguiente manera: «Hacer L1 si se cumple L2 y L3 y ... y Ln».

Veamos un ejemplo:

hija (A, B) <– mujer (A) & padre (B, A).

que se leería de la siguiente manera: «A es hija de B si A es mujer y B es padre de A». En este ejemplo hemos supuesto que los nombres que comienzan por una letra mayúscula representan variables, y pueden sustituirse por cualquier cadena de caracteres que cumpla las condiciones adecuadas. En nuestro caso, si tenemos en cuenta los ejemplos de datos definidos anteriormente, es evidente que podríamos sustituir A por «luisa» y B por «juan», con lo que la instrucción anterior nos quedaría:

hija (luisa, juan) <– mujer (luisa) & padre (juan, luisa)

que se leería así: «Luisa es hija de Juan si Luisa es mujer y Juan es padre de Luisa». Pero como ya sabemos que Luisa es mujer y que Juan es padre de Luisa (pues ya lo definimos un poco más arriba) la conjunción de dichos datos y de esta instrucción permite al intérprete o compilador de PROLOG deducir que, en efecto, Luisa es hija de Juan.

Naturalmente, la definición anterior de la relación «hija» es incompleta. Una persona puede ser hija de otra, no sólo si ésta es su padre, sino también si es su madre. Para mantener esto en cuenta en PROLOG, bastaría con introducir la siguiente instrucción adicional:

hija (A, B) <– mujer (A) & madre (B, A).

que se añadiría a la anterior. El orden en que se escriban estas instrucciones, en principio, no importa, pues ya hemos dicho que PROLOG es un lenguaje no procedimental.

Vamos a ver, como ejemplo de programación en PROLOG, cómo se resolvería en este lenguaje el problema lógico que propusimos en el capítulo anterior, cuyo enunciado vamos a repetir aquí. Dicho problema está sacado del libro *¿Cómo se llama este libro?,* de Raymond Smullyan (Ediciones Cátedra, 1978). El problema se enunica así:

«Cuando Alicia entró en el bosque del olvido no lo olvidó todo, solamente ciertas cosas. A menudo olvidaba su nombre, y una de las cosas que más disposición tenía a olvidar era el día de la semana. Ahora bien, el León y el Unicornio visitaban frecuentemente el bosque. Los dos son criaturas extrañas. El León miente los lunes, martes y miércoles y dice la verdad los otros días de la semana. El Unicornio, por otra parte, miente los jueves, viernes y sábados, pero dice la verdad los restantes días de la semana.»

«Un día Alicia se encontró con el León y el Unicornio que descansaban bajo un árbol. Ellos dijeron lo siguiente»:

«León: Ayer fue uno de los días en que me toca mentir.»

«Unicornio: Ayer fue uno de los días en que me toca mentir.»

«A partir de estas dos frases, Alicia (que era una chica lista) fue capaz de deducir el día de la semana. ¿Qué día era éste?

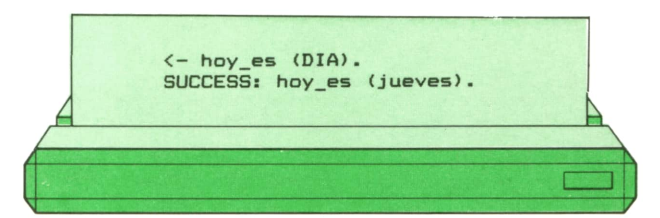
La solución en PROLOG es la siguiente:

```
ayer (domingo, sábado).
ayer (lunes, domingo).
ayer (martes, lunes).
ayer (miércoles, martes).
ayer (jueves, miércoles).
ayer (viernes, jueves).
ayer (sábado, viernes).
león_miente (lunes).
león_miente (martes).
león_miente (miércoles).
unic_miente (jueves).
unic_miente (viernes).
unic_miente (sábado).
león_habla (DIA) <- ayer (DIA, AYER)
                 & león_miente (DIA)
               & ¬ león_miente (AYER).
león_habla (DIA) <- ayer (DIA, AYER)
               & ¬ león_miente (DIA)
                 & león_miente (AYER).
unic_habla (DIA) <- ayer (DIA, AYER)
                 & unic_miente (DIA)
               & ¬ unic_miente (AYER).
unic_habla (DIA) <- ayer (DIA, AYER)
               & ¬ unic_miente (DIA)
                 & unic_miente (AYER).
hoy_es (DIA) <- león_habla (DIA)
              & unic_habla (DIA).
```

En este programa hemos supuesto que los nombres que comienzan por una letra mayúscula representan variables. El signo ¬ representa la negación lógica. Puede observarse lo siguiente:

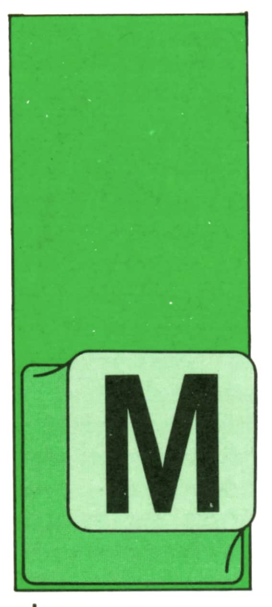
1. Las siete primeras instrucciones son datos que definen cuál es el día de la semana anterior a cada día concreto.
2. Las tres instrucciones siguientes son también datos que dicen qué días miente el león.
3. Las tres instrucciones siguientes dicen qué días miente el unicornio.
4. Las dos instrucciones siguientes le indican al programa que el león sólo puede decir esa frase el día que miente, si el día anterior no mintió, o el día que no miente, si el día anterior mintió.
5. Las dos instrucciones siguientes proporcionan la misma información respecto al unicornio.
6. La última instrucción nos dice que hoy tiene que ser el día en que tanto el león como el unicornio pudieron pronunciar sus frases respectivas.

Veamos cómo se resuelve el problema lógico ejecutando el programa anterior:



APLICACIONES

OTROS PROGRAMAS



MUCHAS de las tareas que se realizan habitualmente son fácilmente mecanizables, si bien no se pueden incluir en ninguno de los programas que hemos visto hasta ahora.

Realizaremos un recorrido por todos aquellos programas difícilmente clasificables, pero que brindarán una considerable ayuda a quien los utilice.

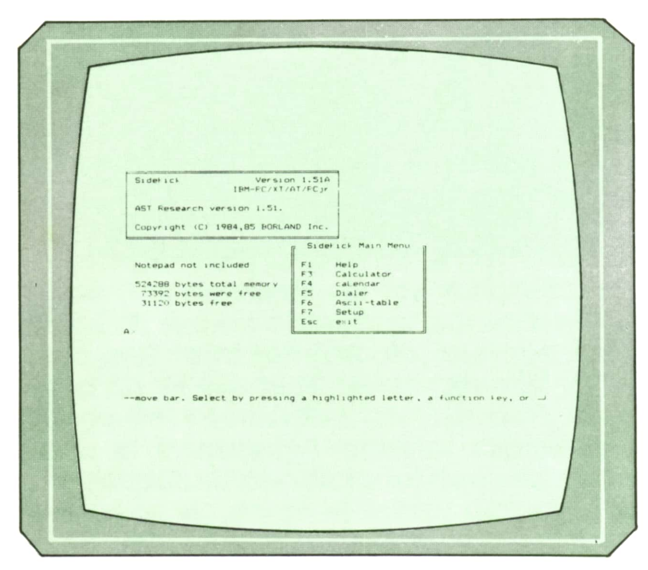
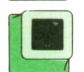




Fig. 1. Programa Siderick.



Agendas electrónicas

En este apartado se incluyen todos aquellos programas que de una forma u otra permiten planificar la actividad a corto y medio plazo, además de permitir llevar un fichero tipo agenda telefónica, que permitirá buscar datos sobre teléfonos o direcciones de forma rápida con opciones de marcado automático de números de teléfono, para aquellos ordenadores conectados a un modem.

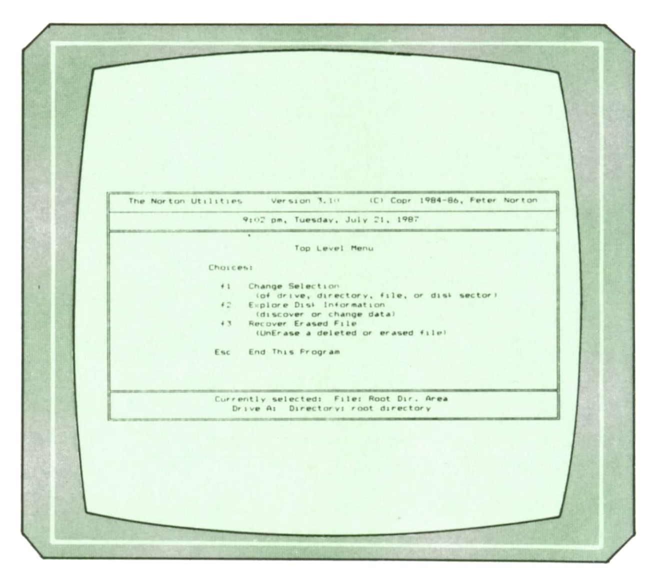




Fig. 2. Programa de edición de disco.

Algunos programas tienen otras opciones, como el block de notas, que permite almacenar datos sin demasiada relación entre ellos, pero siempre necesarios. La mayor parte de ellos incluyen también una calculadora para realizar operaciones matemáticas, y calendario, que puede ser perpetuo o no.

Este tipo de programas suelen estar en un segundo plano en la memoria, permitiendo la ejecución normal de otros, llamando a los primeros por la pulsación de una serie de teclas.

Sin duda, el más conocido de estos programas es el SIDEKICK, que permite todas las opciones anteriormente dichas.



Comunicaciones

De la comunicación entre el ordenador y el exterior se encargan los programas de comunicaciones; éstos permiten transmitir datos, en forma de ficheros generalmente, a otros ordenadores o re-

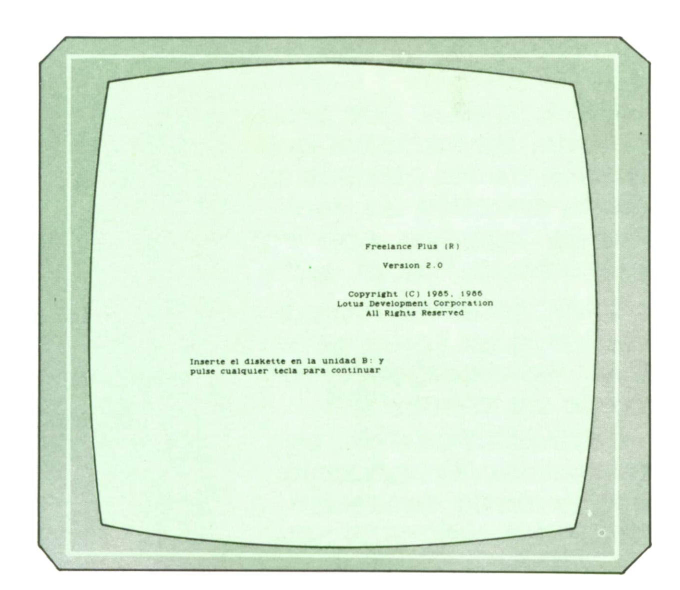




Fig. 3. Pantalla de acceso a Lotus Freelance.

des, bien sea a través de una red local o de una línea telefónica.

Entre los programas más conocidos de comunicaciones están:

NETWARE DE NOVELL: Comunicaciones entre micros en red local, multiusuario y multitarea.

SSI5251: Programa que, junto a algunos complementos de hardware, permite la comunicación entre un ordenador PC y un sistema 36 ó 38 de IBM.

IBM PC NETWORK PROGRAM: Software de red que permite dos modos de funcionamiento, para principiantes y expertos.

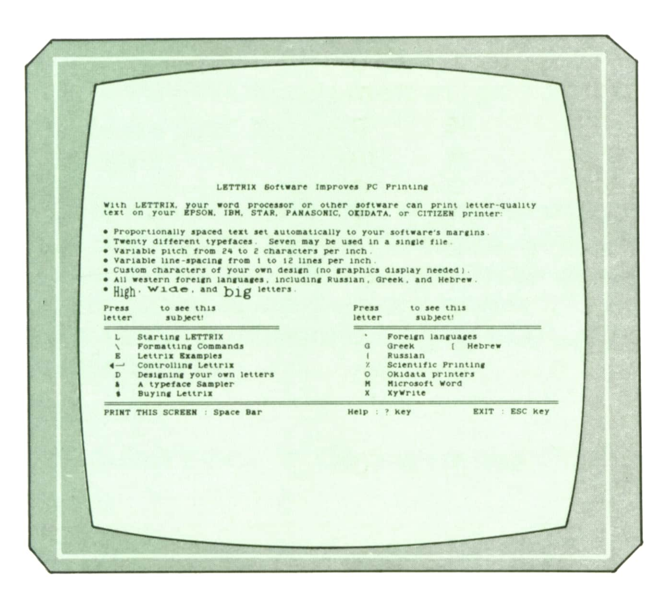




Fig. 4. Programa para diseño de letras.



Utilidades

Dentro de este grupo de programas se engloban todos aquellos cuya inclusión no se puede hacer en los grupos de programas anteriores; generalmente suelen proporcionar una herramienta de ayuda al usuario, y se orientan a la solución de los más diversos problemas. Un primer tipo de programas de utilidades son aquellos con los que se pueden manejar la información del disco directamente, bien sector por sector o por ficheros, permitiendo de este modo realizar cambios directamente en cualquier sector del disco. Una importante faceta de estos programas es la posibilidad de recuperar ficheros borrados siempre y cuando no se grabe posteriormente a su borrado nada en el disco. El más conocido de todos es, sin duda, el programa NORTON UTILITIES, que permite la edición del disco byte por byte.

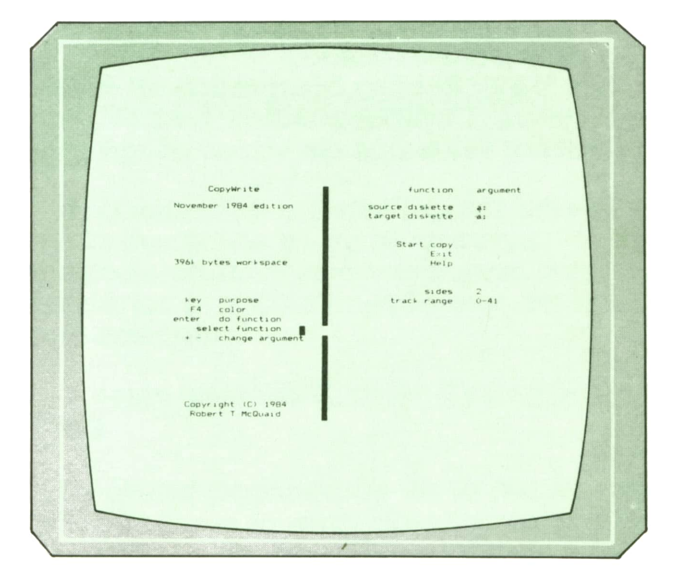




Fig. 5. Programa copiador.

Los programas de protección de ficheros también se pueden incluir dentro de las utilidades; estos programas tienen como misión fundamental la protección de programas en disquette contra las copias no autorizadas. SYSPRO constituye uno de esos programas de protección del software, de fácil uso, permite proteger ficheros con una sola instrucción.

POLICE es un sistema de protección del software diseñado para proteger los programas residentes en disco contra copias no autorizadas.

Otro tipo de utilidades son aquellas que trabajan con ficheros y van desde el simple programa de ordenación hasta los complejos programas de diseño de aplicaciones.

MENUMAKER es un programa que permite la creación de menús para su utilización por otros programas; a cada menú se le pueden asignar palabras clave; se pueden incluir hasta 26 menús en un disco.

MICROSOFT CLOUT: Permite el análisis y consulta de ficheros en un inglés tipo coloquial interpreta sin problemas ficheros procedentes de los programas Multiplan, Lotus, Base, Ascii, etc.

SUPERSORT: Programa que clasifica, intercala y selecciona información de los ficheros de datos.

GAPE: Programa generador de aplicaciones, crea programas en lenguaje basic compilable.

SORT 1.0: Clasifica registros de ficheros ascendente o descendentemente, fusiona varios ficheros, así como selecciona registros para excluirlos.

TOP VIEW: Realiza búsquedas en bases de datos. Compara datos permitiendo presentar los datos de varios programas en forma de ventanas que pueden cambiar de aspecto y posición.

DATA SCREEN: Este programa es un generador de pantallas que permite la definición de los campos de entrada. Facilita la creación de ventanas de trabajo. Puede trabajar con varios lenguajes como Basic, Pascal, etc.

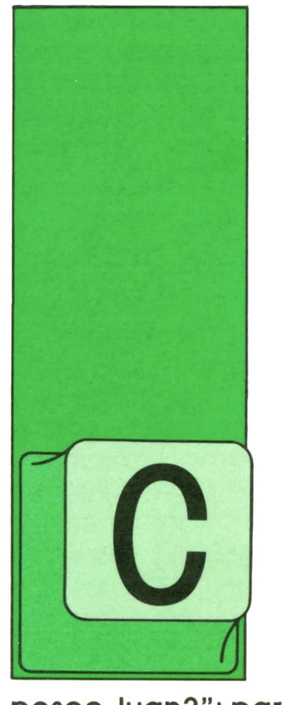
BASIC WINDOW: Programa de diseño de ventanas en Basic; genera una rutina en Basic para cargar ventanas en cualquier punto de la pantalla.

REVELATION: Este programa incluye un generador de aplicaciones, un compilador de Basic avanzado, un conjunto de utilidades. Asimismo soporta redes locales y comunicaciones.

FIXED DISK ORGANIZER: Facilita la organización de la información contenida en un disco fijo.

Los programas que hemos presentado aquí constituyen una pequeña parcela dentro de la gran cantidad de programas de ayuda que existen en el mercado; es casi seguro que cualquier tarea que realice un usuario normal de ordenadores personales encuentre un programa que se adapte a sus necesidades.

OTROS LENGUAJES





PROLOG-2

CON lo visto hasta ahora podemos declarar hechos y hacer preguntas que nos digan si algunos hechos son ciertos o falsos. Sería útil hacer preguntas del tipo "¿qué cosas posee Juan?"; para esto existen las variables.

Las variables son objeto que pueden tomar cualquier valor. Para distinguirlos el intérprete de PROLOG tomará como variable todo nombre que comience con una letra mayúscula, por lo que podrá ser cualquier nombre. Cuando realizamos una pregunta con una variable el intérprete comenzará a buscar en su base de datos todos los objetos que verifican dicha relación.

Por ejemplo, si con la siguiente base de conocimientos:

```
posee (juan, libro).
posee (juan, casa).
posee (pedro, lapiz).
```

Al preguntarle:

```
?- posee (juan, X).
X = libro.
```

Sólo nos contesta con el primer objeto que encuentra que verifica la relación. Si quisiéramos que continuase deberíamos pulsar la tecla de punto y coma ";". De esta forma continuaría con la búsqueda encontrando:

```
X = casa
```

Si pulsáramos ";" de nuevo nos diría:

```
No
```

indicando que en su base de datos no existen más objetos que verifiquen la relación.



Conjunciones

Para realizar preguntas más complejas e interesantes debemos ser capaces de unir distintas proposiciones mediante conjunciones.

La conjunción y (*and* en inglés) se representa mediante la coma ",". La proposición total será verdad únicamente en el caso que todas las proposiciones que la componen son ciertas. Suponiendo la base de conocimientos del anterior ejemplo veamos otro:

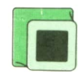
```
?- posee (pedro,libro), posee (pedro,casa).
YES
?- posee (pedro,libro), posee (pedro,lapiz)
NO
```

La conjunción o (*or* en inglés) se representa mediante el punto y coma ";". La proposición total será verdadera si alguna de las proposiciones lo son. Veamos otro ejemplo:

```
?- posee (pedro,libro); posee (pedro,lapiz).
YES
```

La última proposición es la negación. Esta hace que lo verdadero sea falso y viceversa. La función sobre la que actúa deberá ir entre paréntesis.

```
?- NOT (posee (pedro,libro)).
NO
?- posee (pedro,libro), NOT (posee (pedro,lapiz)).
YES
```



Reglas

Existen casos en que declarar gran cantidad de hechos puede ser largo y pesado. Para estos casos existen las re-

glas que indican formas de obtener nuevas relaciones a partir de las ya existentes. En castellano son equivalentes a las construcciones del tipo "si... entonces...". Corresponde a la implicación de la lógica formal.

Algunas de las diferentes formas en que se pueden escribir las reglas son:

```
tio (x,z) : -padre (x,y), hermano (y,z).
tio (x,z) <- padre (x,y) & hermano (y,z).
tio (-x,-z):- padre (-x,-y), hermano (-y,-z).
+ tio (*x,*z) - padre (*z,*y), hermano (*y,*z).
((tio x1 x3) (padre x1 x2) (hermano x2 x3).
tio (x,z): padre (x,y); hermano (y,z).
```

En el siguiente capítulo veremos algunos ejemplos de aplicación de reglas.

PROLOG-3

Además de las operaciones lógicas, PROLOG también posee operaciones numéricas. Para realizar operaciones entre números enteros pueden usarse las típicas operaciones (*,+,−,/) y además MOD para el resto de la división entera. La notación para este tipo de operaciones es más libre que en otro tipo de funciones, pudiéndose usarse cualquiera de las siguientes:

— Notación infija: El operador se escribe entre los operadores, es la notación usada normalmente:

```
x + (y * z)
```

— Notación prefija: También conocida como notación polaca, por ser desarrollada por el lógico polaco Jan Lukasiewicz, pero lo complejo de su nombre evitó ser conocido por 1. Esta notación es la utilizada por el LISP. En ella el operador precede a los operandos sobre los que actúa.

```
+ (x, *(y,z))
```

Para realizar comparaciones entre números existen las siguientes funciones:

```
x = y     x igual a y
x \= y    x distinto de y
x < y     x menor que y
x > y     x mayor que y
x =< y    x menor o igual que y
x >= y    x mayor o igual que y
```

En los operadores que están formados por dos caracteres no puede cambiarse el orden, como en otros lenguajes, por eso los detallamos todos.

Por estar basado en el LISP también incorpora el manejo de listas, con la mayoría de las funciones de éste. Las listas se representan de una forma un poco distinta, por ejemplo, la lista con un único elemento se representaría como:

Donde el operador punto «.» es el que sirve para unir los elementos de las listas. Existen las mismas funciones que se vieron en LISP para el manejo de listas.

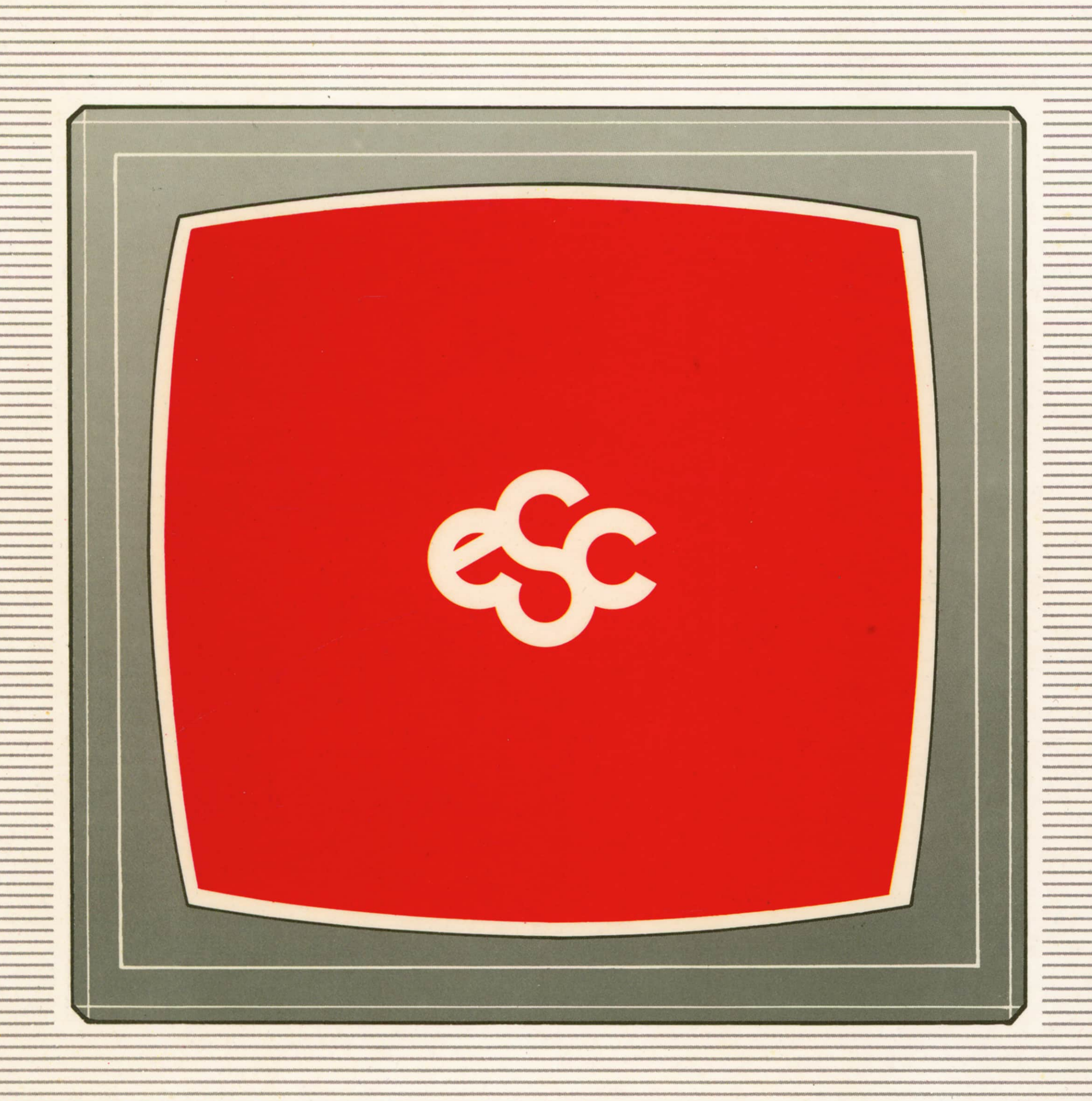
En el siguiente ejemplo vemos un uso de éstas en un programa que calcula la derivada simbólica de una expresión:

```
d(x,x,1)  :-  !.
d(c,x,0)  :-  atomic(c).
d(u+v,x,a+b)  :-  d(u,x,a), d(v,x,b).
d(u−v,x,u−v)  :-  d(u,x,a), d(v,x,b).
d(c*u,x,c*a)  :-  atomic (c), c \= x, d(u,x,a), !.
d(u^v,x,v*w*u^(v−1)  :-  atomic (v), c \+ x,d(u,x,v).
d(log(u),x,a*u^(−1))  :-  d(u,x,a).
```

En el ejemplo final vemos, con una notación diferente, pero la más usada por los intérpretes de PROLOG un ejemplo de utilización de reglas para construir una compleja base de datos.

```
((Hembra Isabel-de-York))
((Hembra Catalina))
((Hembra Maria))
((Hembra Ana))
((Hembra Juana))
((Hembra Isabel))
((Varón Enrique7))
((Varón Enrique8))
((Varón Eduardo))
((madre Isabel-de-York Enrique8))
((madre Catalina María))
((madre Ana Isabel))
((madre Juana Eduardo))
((padre Enrique7 Enrique8))
((padre Enrique8 Maria))
((padre Enrique8 Isabel))
((padre Enrique8 Eduardo))
((hijo X Y)
  (varón X)
    (padre Y X))
((hijo X Y)
  (varón X)
  (madre Y X))
((hija X Y)
  (mujer X)
  (padre Y X))
((hija X Y)
  (mujer X)
  (madre Y X))
```

```
((nieto X Y)
  (varón X)
  (abuelo Y X))
((nieto X Y)
  (varón X)
  (Abuela Y X))
((nieta X Y)
  (mujer X)
  (abuelo Y X))
((nieta X Y)
  (mujer X)
  (abuela Y X))
((abuelo X Y)
  (padre X Z)
  (padre Z Y))
((abuelo X Y)
  (padre X Z)
  (madre Z Y))
((abuela X Y)
  (madre X Z)
  (padre Z Y))
((abuela X Y)
  (madre X Z)
  (madre Z Y))
((igual_padres X Y)
  (igual_padre X Y)
  (igual_madre X Y))
((igual_madre X Y)
  (madre Z X)
  (madre Z Y))
((igual_padre X Y)
  (padre Z X)
  (padre Z Y))
((hermano X Y)
  (varón X)
  (igual_padres X Y)
  (NOT ? ((EQ X Y))))
((igual X X))
((hermana X Y)
  (mujer X)
  (igual_padres X Y)
  (NOT ? ((EQ X Y))))
((antepasado X Y)
  (padre X Y))
((antepasado X Y)
  (madre X Y))
((antepasado X Y)
  (padre X Z)
  (antepasado X Y))
((antepasado X Y)
  (madre X Z)
  (antepasado Z Y))
((descendiente X Y)
  (antepasado Y X))
((primo X Y)
  (hijo X Z)
  (hermano Z W)
  (padre W Y))
((primo X Y)
  (hijo X Z)
  (hermano Z W)
  (madre W Y))
((primo X Y)
  (hijo X Z)
  (hermana Z W)
  (padre W Y))
((primo X Y)
  (hijo X Z)
  (hermana Z W)
  (madre W Y))
((prima X Y)
  (hija X Z)
  (hermano Z W)
  (padre W Y))
((prima X Y)
  (hija X Z)
  (hermano Z W)
  (madre W Y))
((prima X Y)
  (hija X Z)
  (hermana Z W)
  (padre W Y))
((prima X Y)
  (hija X Z)
  (hermana Z W)
  (madre W Y))
((tío X Y)
  (hermano X Z)
  (padre Z Y))
((tío X Y)
  (hermano X Z)
  (madre Z Y))
((tía X Y)
  (hermana X Z)
  (padre Z Y))
((tía X Y)
  (hermana X Z)
  (madre Z Y))
((sobrino X Y)
  (varón X)
  (tío Y X))
((sobrino X Y)
  (varón X)
  (tía Y X))
((sobrina X Y)
  (mujer X)
  (tío Y X))
((sobrina X Y)
  (mujer X)
  (tía Y X))
```



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼